Using Concurrent Function Evaluations to Identify Local Minima of a Derivative-free Optimization Problem

Jeffrey Larson Stefan Wild

Argonne National Laboratory

August 12, 2015

Motivation

- We want to identify distinct, “high-quality”, local minimizers of

$$\begin{aligned} \text{minimize } & f(x) \\ l \leq x & \leq u \\ x & \in \mathbb{R}^n \end{aligned}$$

- High-quality can be measured by more than the objective.

Motivation

- We want to identify distinct, "high-quality", local minimizers of

$$\begin{aligned} \text{minimize } & f(x) \\ l \leq x & \leq u \\ x & \in \mathbb{R}^n \end{aligned}$$

- High-quality can be measured by more than the objective.
- Derivatives of f may or may not be available.

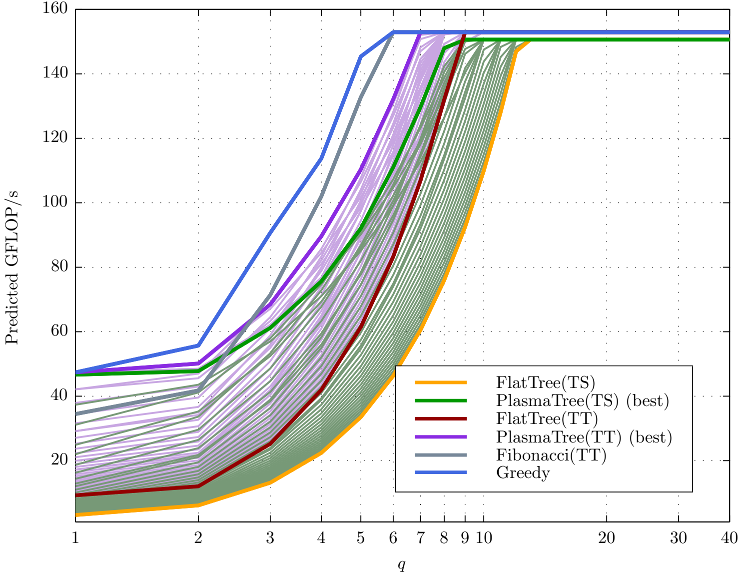
Motivation

- We want to identify distinct, “high-quality”, local minimizers of

$$\begin{aligned} \text{minimize}\, f(x) \\ l \leq x \leq u \\ x \in \mathbb{R}^n \end{aligned}$$

- High-quality can be measured by more than the objective.
- Derivatives of f may or may not be available.
- The simulation f is likely using parallel resources, but it does not utilize the entire machine.

Why concurrency? Tiled QR example



[Bouwmeester, et al., Tiled QR Factorization Algorithms, 2011]

Global optimization is difficult

Theorem (Törn and Žilinskas, *Global Optimization*, 1989)

An algorithm converges to the global minimum of any continuous f on a domain $\mathcal{D}$ if and only if the algorithm generates iterates that are dense in $\mathcal{D}$.

Global optimization is difficult

Theorem (Törn and Žilinskas, *Global Optimization*, 1989)

An algorithm converges to the global minimum of any continuous f on a domain $\mathcal{D}$ if and only if the algorithm generates iterates that are dense in $\mathcal{D}$.

- Either assume additional properties about the problem
  - convex f
  - separable f
  - finite domain $\mathcal{D}$

Global optimization is difficult

Theorem (Törn and Žilinskas, *Global Optimization*, 1989)

An algorithm converges to the global minimum of any continuous f on a domain $\mathcal{D}$ if and only if the algorithm generates iterates that are dense in $\mathcal{D}$.

- Either assume additional properties about the problem
  - convex f
  - separable f
  - finite domain $\mathcal{D}$

- Or possibly wait a long time (or forever)

Global optimization is difficult

Theorem (Törn and Žilinskas, *Global Optimization*, 1989)

An algorithm converges to the global minimum of any continuous f on a domain $\mathcal{D}$ if and only if the algorithm generates iterates that are dense in $\mathcal{D}$.

- Either assume additional properties about the problem
  - convex f
  - separable f
  - finite domain $\mathcal{D}$
  - concurrent evaluations of f
- Or possibly wait a long time (or forever)

Global optimization is difficult

Theorem (Törn and Žilinskas, *Global Optimization*, 1989)

An algorithm converges to the global minimum of any continuous f on a domain $\mathcal{D}$ if and only if the algorithm generates iterates that are dense in $\mathcal{D}$.

- Either assume additional properties about the problem
  - convex f
  - separable f
  - finite domain $\mathcal{D}$
  - concurrent evaluations of f
- Or possibly wait a long time (or forever)

The theory can be more than merely checking that a method generates iterates which are dense in the domain.

Multistart Methods

Given some local optimization routine $\mathcal{L}$:

Algorithm 1: General Multistart

for $k = 1, 2, \ldots$ **do**

- Evaluate f at N points drawn from $\mathcal{D}$
- Start $\mathcal{L}$ at some set (possibly empty) of previously evaluated points

Multistart Methods

Given some local optimization routine $\mathcal{L}$:

Algorithm 1: General Multistart

for $k = 1, 2, \ldots$ **do**

  Evaluate f at N points drawn from $\mathcal{D}$

  Start $\mathcal{L}$ at some set (possibly empty) of previously evaluated points

- Get to use problem specific local optimization routines.
- Possibly multiple levels of parallelism (objective, local method, global method); $\mathcal{L}$ may involve many sequential evaluations of f...

Multistart Methods

Given some local optimization routine $\mathcal{L}$:

Algorithm 1: General Multistart

for $k = 1, 2, \ldots$ **do**
  Evaluate f at N points drawn from $\mathcal{D}$
  Start $\mathcal{L}$ at some set (possibly empty) of previously evaluated points

- Get to use problem specific local optimization routines.
- Possibly multiple levels of parallelism (objective, local method, global method); $\mathcal{L}$ may involve many sequential evaluations of f...

- Which points should start runs?
- If resources are limited, how should points from each run receive priority?

Multistart Methods

Given some local optimization routine $\mathcal{L}$:

Algorithm 1: General Multistart

for $k = 1, 2, \ldots$ **do**
- Evaluate f at N points drawn from $\mathcal{D}$
- Start $\mathcal{L}$ at some set (possibly empty) of previously evaluated points

- Get to use problem specific local optimization routines.
- Possibly multiple levels of parallelism (objective, local method, global method); $\mathcal{L}$ may involve many sequential evaluations of f...

- Which points should start runs?
- If resources are limited, how should points from each run receive priority?

- Ideally, only one run is started for each minima.
- Exploring by sampling. Refining with $\mathcal{L}$.

Multi-Level Single Linkage

Given some local optimization routine $\mathcal{L}$:

Algorithm 2: MLSL

for $k = 1, 2, \ldots$ **do**

Sample f at N random points drawn uniformly from $\mathcal{D}$

Start $\mathcal{L}$ at all sample points x:

- that has yet to start a run
- $\nexists x_i : \|x - x_i\| \leq r_k$ and $f(x_i) < f(x)$

[Rinnooy Kan and Timmer, *Mathematical Programming*, 39(1):57–78, 1987]

Multi-Level Single Linkage

Given some local optimization routine $\mathcal{L}$:

Algorithm 2: MLSL

---

for $k = 1, 2, \ldots$ **do**

Sample f at N random points drawn uniformly from $\mathcal{D}$

Start $\mathcal{L}$ at all sample points x:

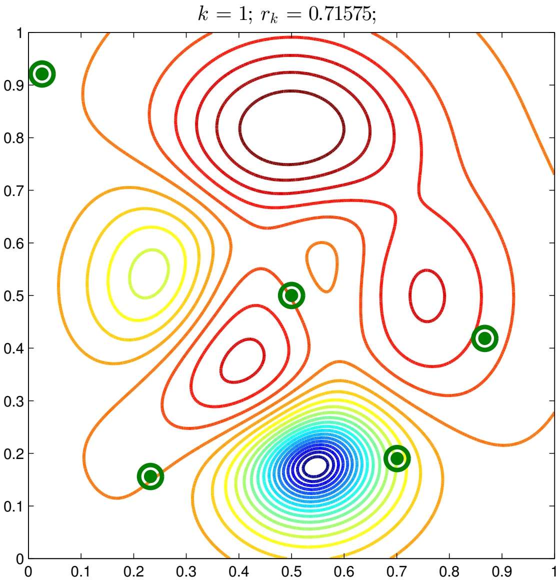
- that has yet to start a run
- $\nexists x_i : \|x - x_i\| \leq r_k$ and $f(x_i) < f(x)$

[Rinnooy Kan and Timmer, *Mathematical Programming*, 39(1):57–78, 1987]

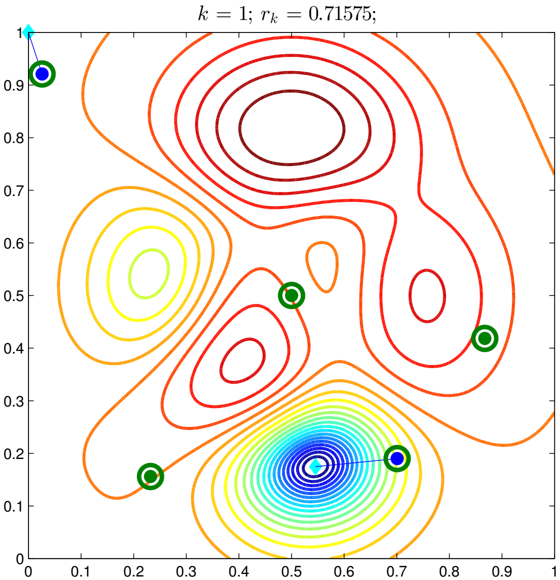
- Doesn't naturally translate when evaluations of f are limited

- Ignores some points when deciding where to start $\mathcal{L}$

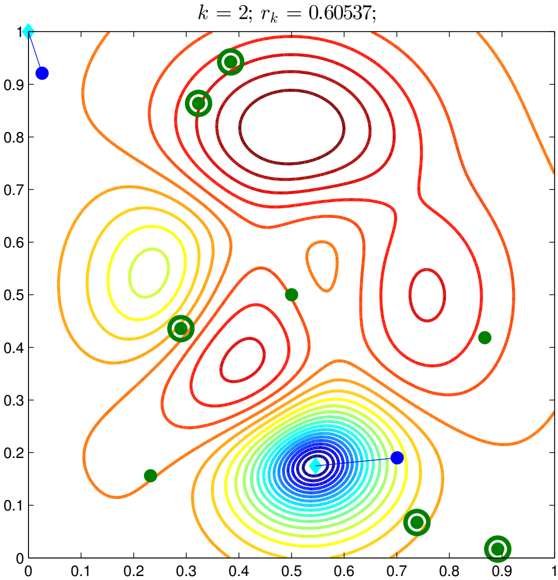
Multi-Level Single Linkage



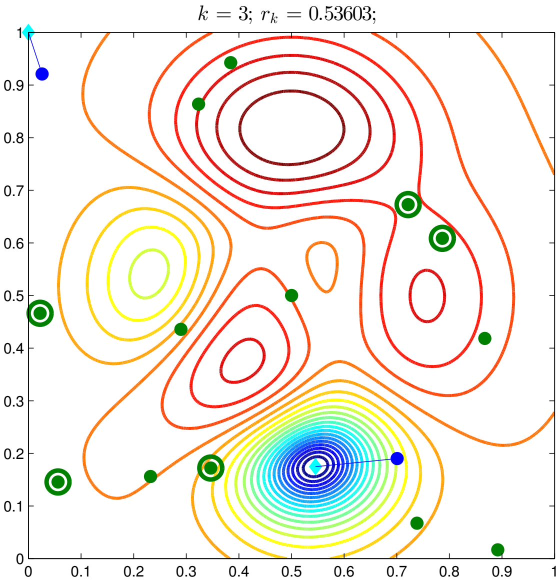
Multi-Level Single Linkage



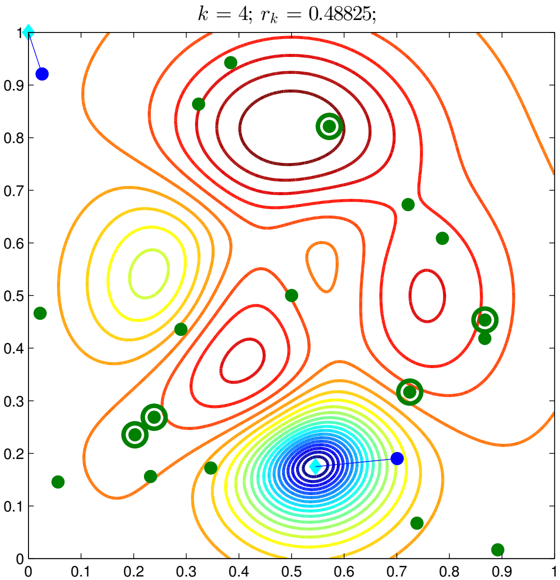
Multi-Level Single Linkage



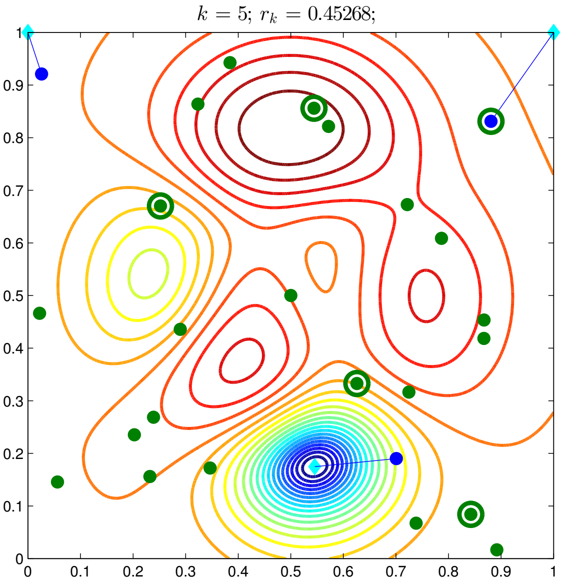
Multi-Level Single Linkage



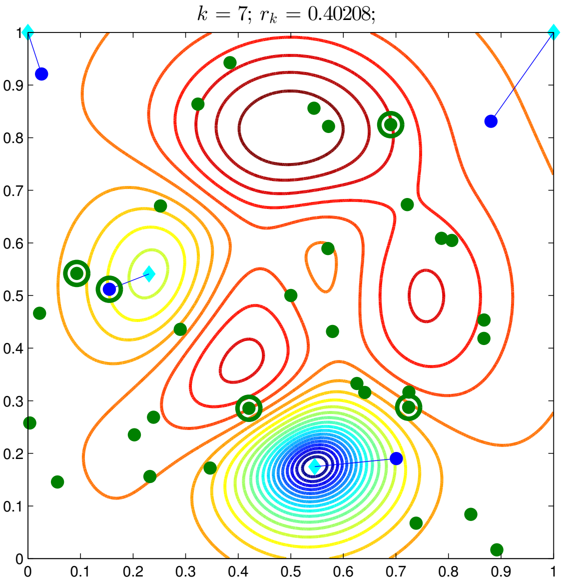
Multi-Level Single Linkage



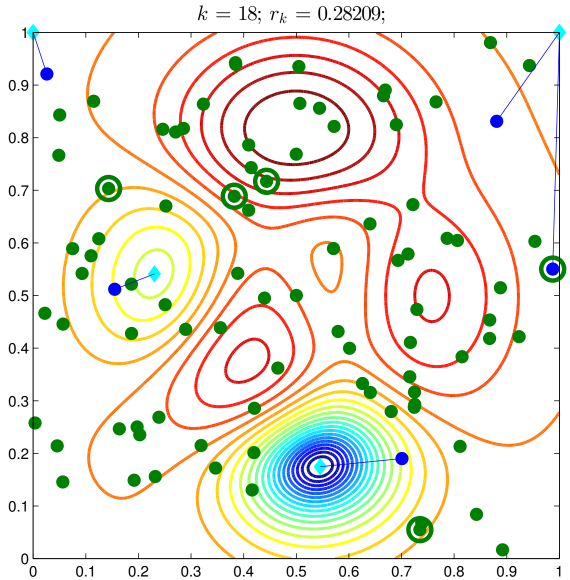
Multi-Level Single Linkage



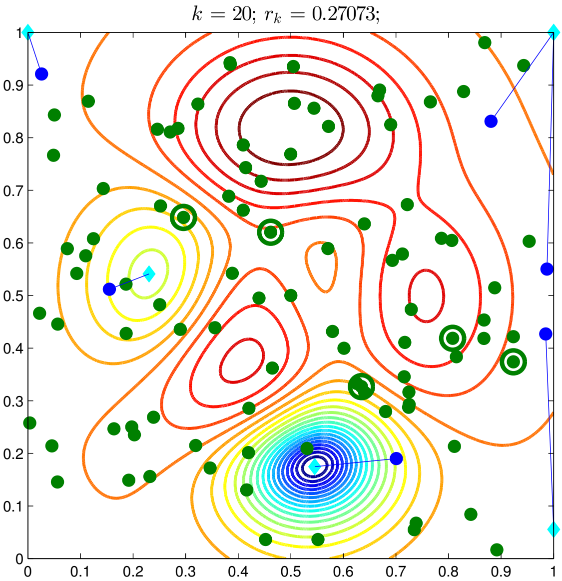
Multi-Level Single Linkage



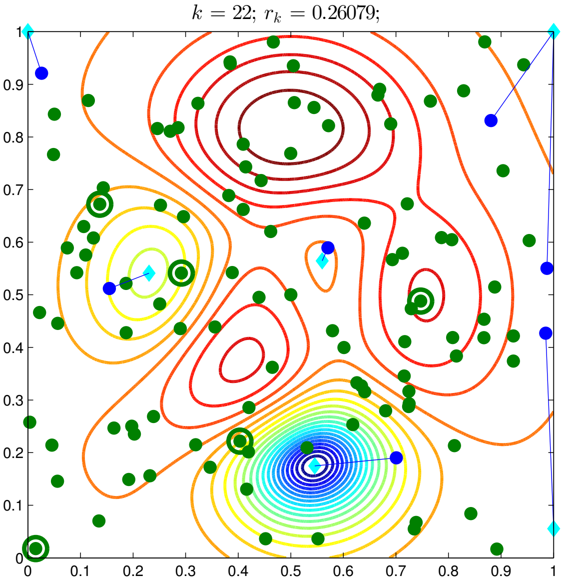
Multi-Level Single Linkage



Multi-Level Single Linkage



Multi-Level Single Linkage



Multi-Level Single Linkage

- $f \in C^2$, with local minima in the interior of $\mathcal{D}$, and the distance between these minima is bounded away from zero.
- $\mathcal{L}$ is strictly descent and converges to a minimum (not a stationary point).
- $$r_k = \frac{1}{\sqrt{\pi}} \sqrt[n]{\Gamma\left(1 + \frac{n}{2}\right) \mathrm{vol}\left(\mathcal{D}\right) \frac{\sigma \log kN}{kN}} \tag{1}$$

Theorem

If $r_k \to 0$, all local minima will be found almost surely.

Multi-Level Single Linkage

- $f \in C^2$, with local minima in the interior of $\mathcal{D}$, and the distance between these minima is bounded away from zero.
- $\mathcal{L}$ is strictly descent and converges to a minimum (not a stationary point).
- $$r_k = \frac{1}{\sqrt{\pi}} \sqrt[n]{\Gamma\left(1 + \frac{n}{2}\right) \mathrm{vol}\left(\mathcal{D}\right) \frac{\sigma \log kN}{kN}} \tag{1}$$

Theorem

If $r_k \to 0$, all local minima will be found almost surely.

If r_k is defined by (1) with $\sigma > 4$, even if the sampling continues forever, the total number of local searches started is finite almost surely.

BAMLM

MLSL: (S2)–(S4)

$\hat{x} \in \mathcal{S}_k$

(S2) $\nexists x \in \mathcal{S}_k$ *with* $[\|\hat{x} - x\| \leq r_k$ *and* $f(x) < f(\hat{x})]$

(S3) $\hat{x}$ *has not started a local optimization run*

(S4) $\hat{x}$ *is at least* μ *from* $\partial\mathcal{D}$ *and* ν *from known local minima*

BAMLM

MLSL: (S2)–(S4)

$\hat{x} \in \mathcal{S}_k$

(S1) $\nexists x \in \mathcal{L}_k$ with $[\|\hat{x} - x\| \leq r_k \text{ and } f(x) < f(\hat{x})]$

(S2) $\nexists x \in \mathcal{S}_k$ *with* $[\|\hat{x} - x\| \leq r_k$ *and* $f(x) < f(\hat{x})]$

(S3) $\hat{x}$ *has not started a local optimization run*

(S4) $\hat{x}$ *is at least* μ *from* $\partial\mathcal{D}$ *and* ν *from known local minima*

BAMLM: (S1)–(S4), (L1)–(L6)

$\hat{x} \in \mathcal{L}_k$

(L1) $\nexists x \in \mathcal{L}_k$ $[\|\hat{x} - x\| \leq r_k \text{ and } f(x) < f(\hat{x})]$

(L2) $\nexists x \in \mathcal{S}_k$ with $[\|\hat{x} - x\| \leq r_k \text{ and } f(x) < f(\hat{x})]$

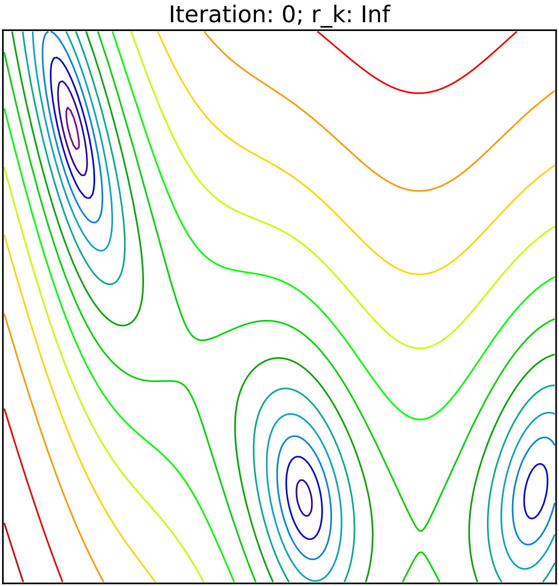
(L3) $\hat{x}$ has not started a local optimization run

(L4) $\hat{x}$ is at least μ from $\partial\mathcal{D}$ and ν from known local minima

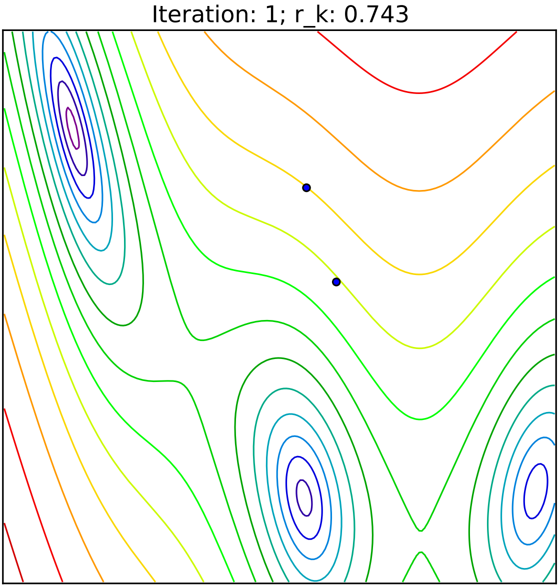
(L5) $\hat{x}$ is not in an active local optimization run and has not been ruled stationary

(L6) $\exists r_k$-descent path in $\mathcal{H}_k$ from some $x \in \mathcal{S}_k$ satisfying (S2-S4) to $\hat{x}$

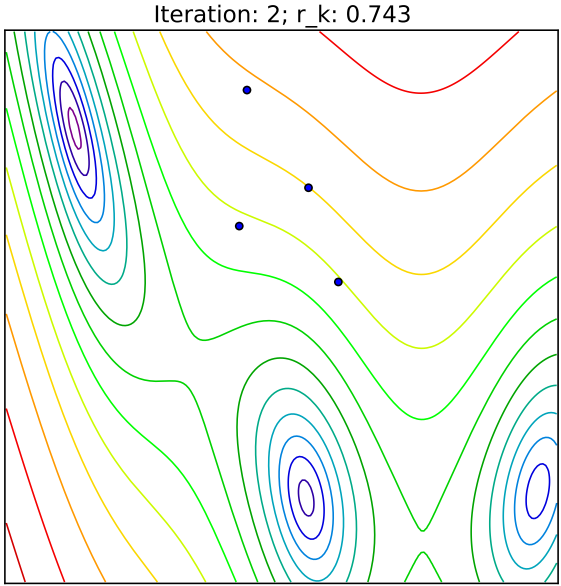
BAMLM



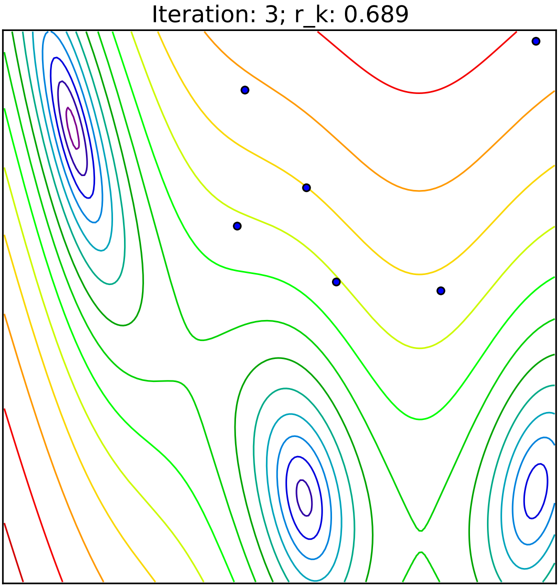
BAMLM



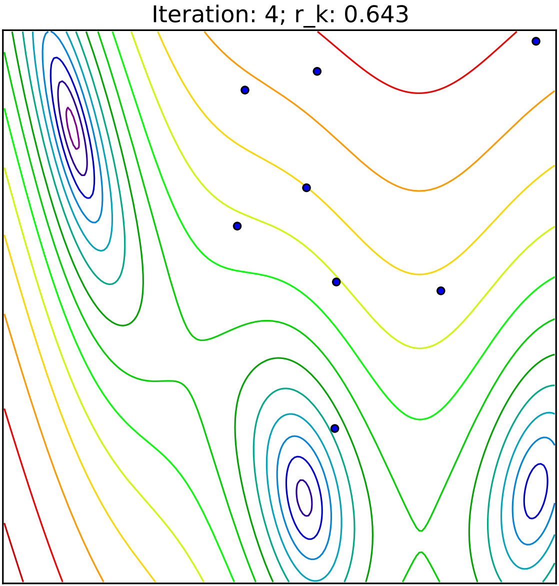
BAMLM



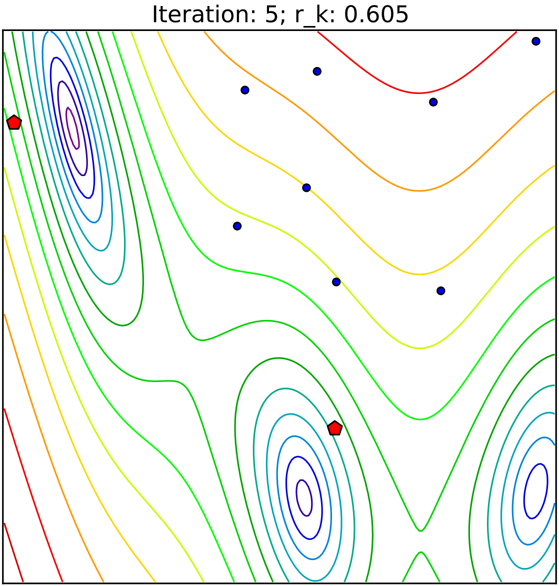
BAMLM



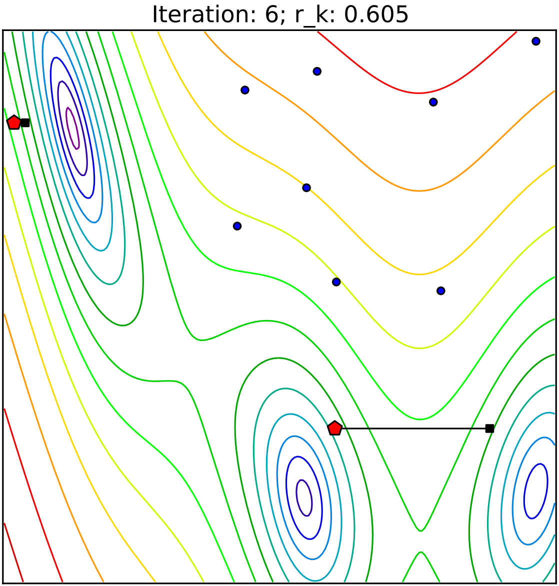
BAMLM



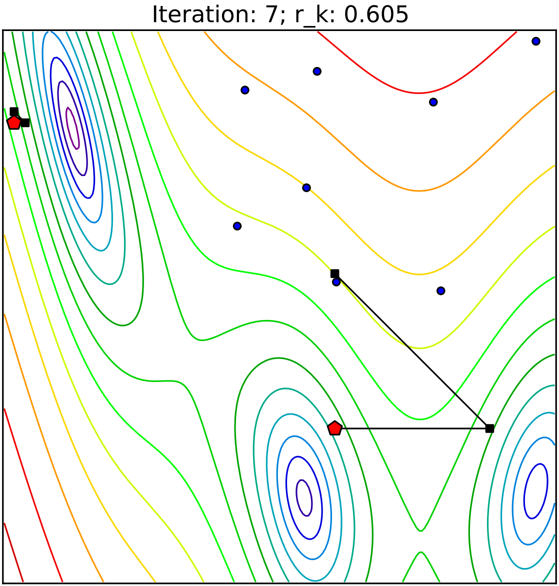
BAMLM



BAMLM

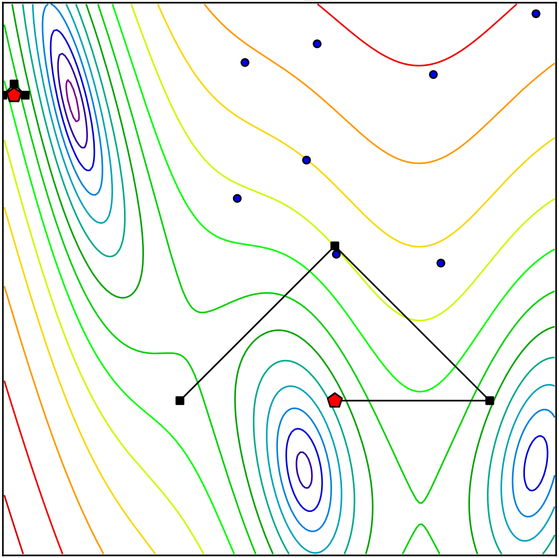


BAMLM



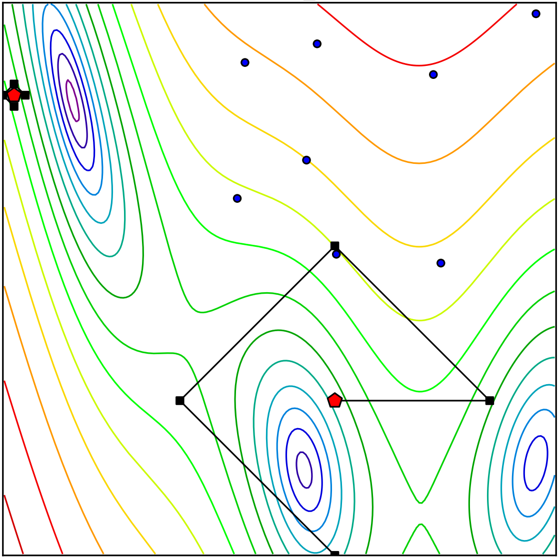
BAMLM

Iteration: 8; r_k: 0.605



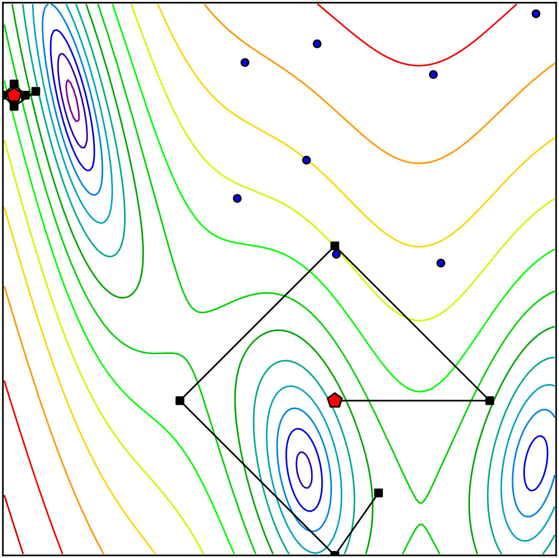
BAMLM

Iteration: 9; r_k: 0.605

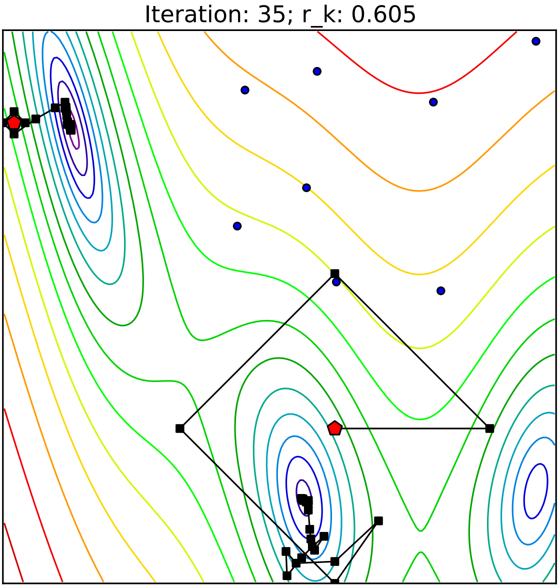


BAMLM

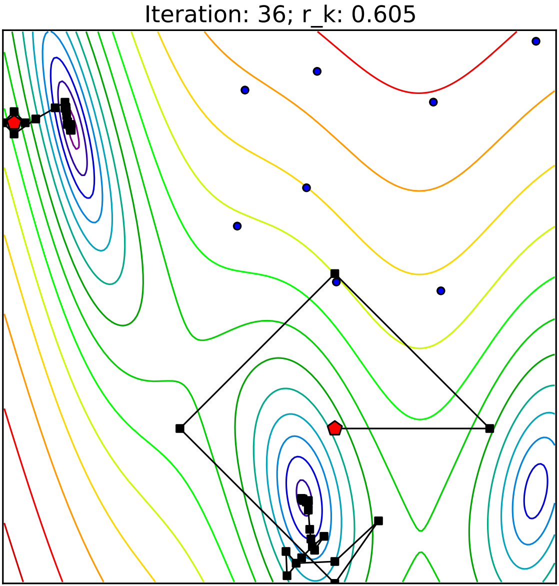
Iteration: 10; r_k: 0.605



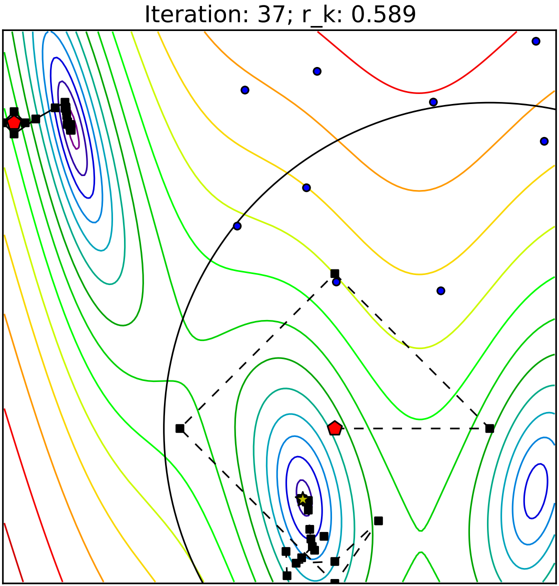
BAMLM



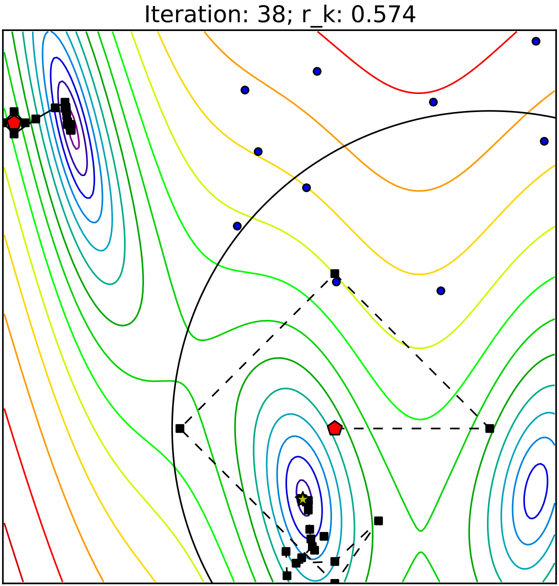
BAMLM



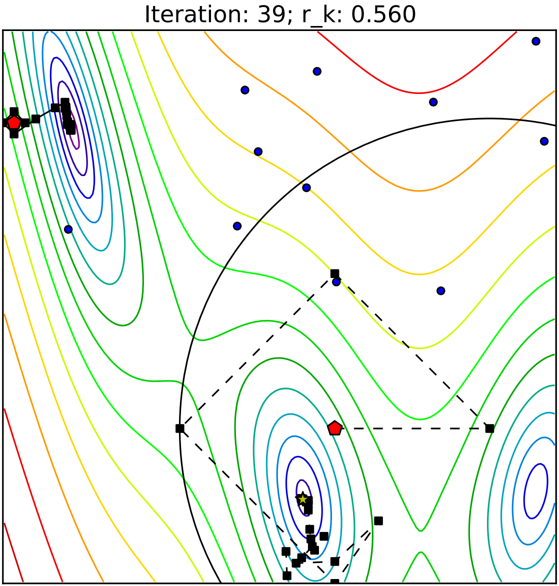
BAMLM



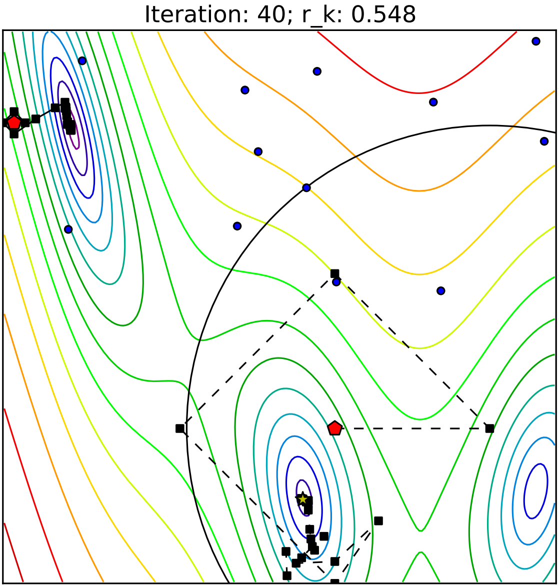
BAMLM



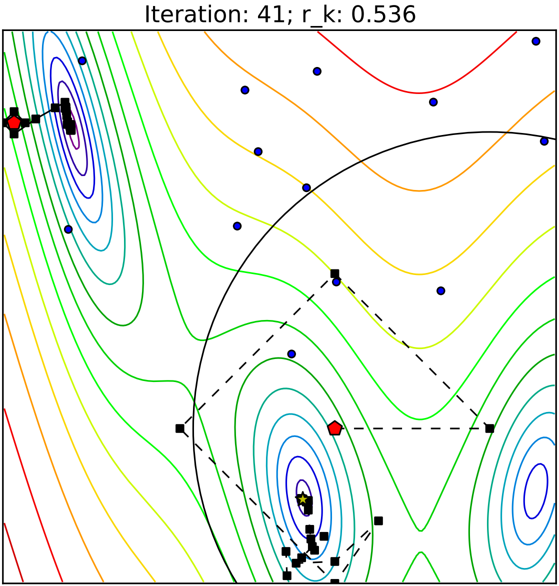
BAMLM



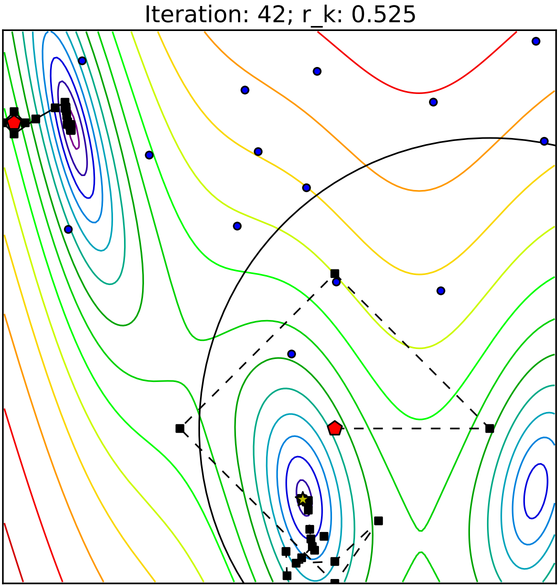
BAMLM



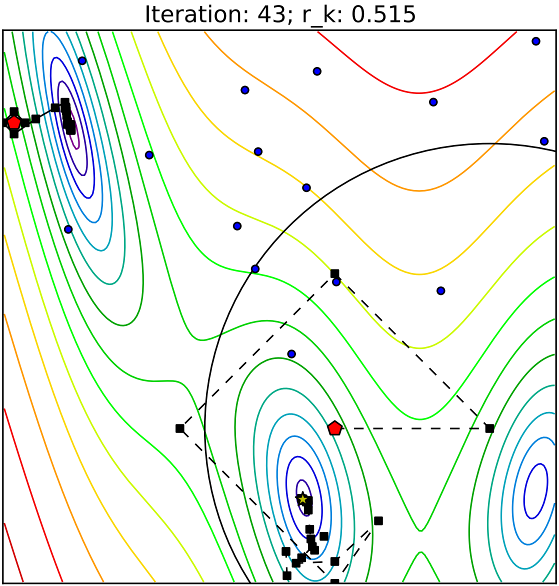
BAMLM



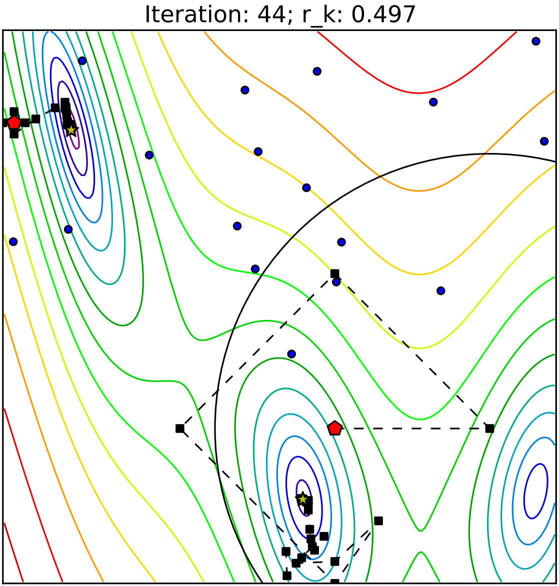
BAMLM



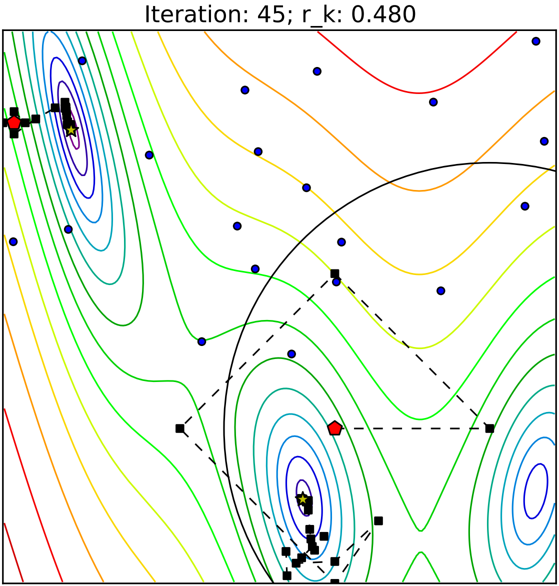
BAMLM



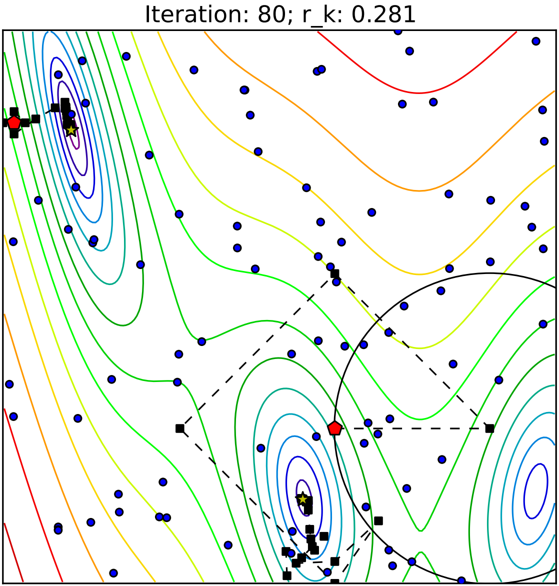
BAMLM



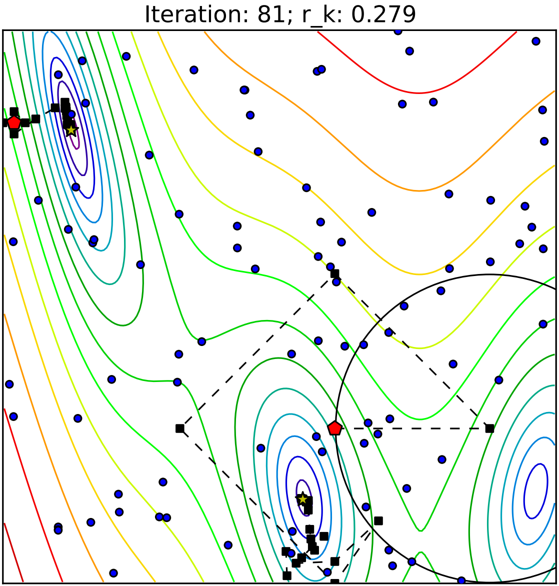
BAMLM



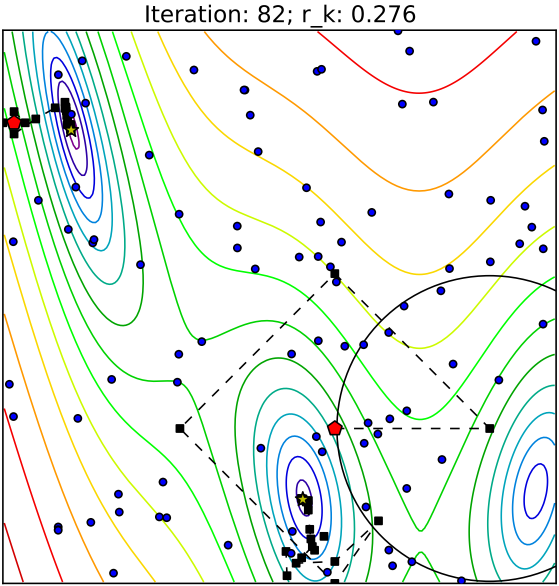
BAMLM



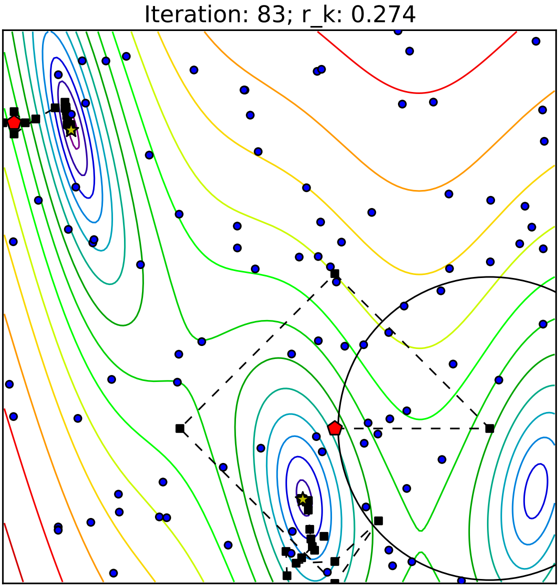
BAMLM



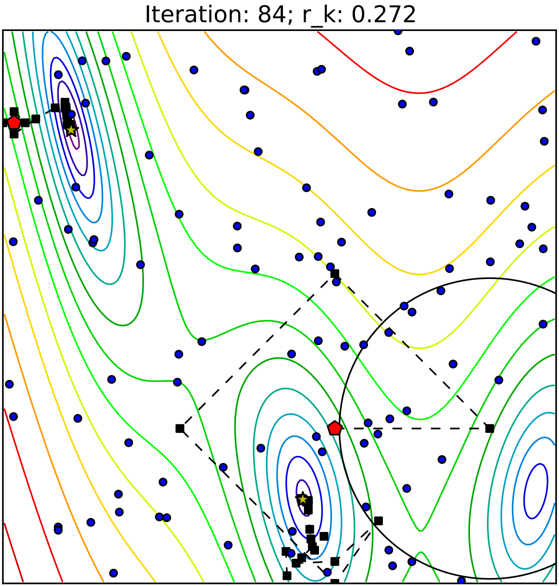
BAMLM



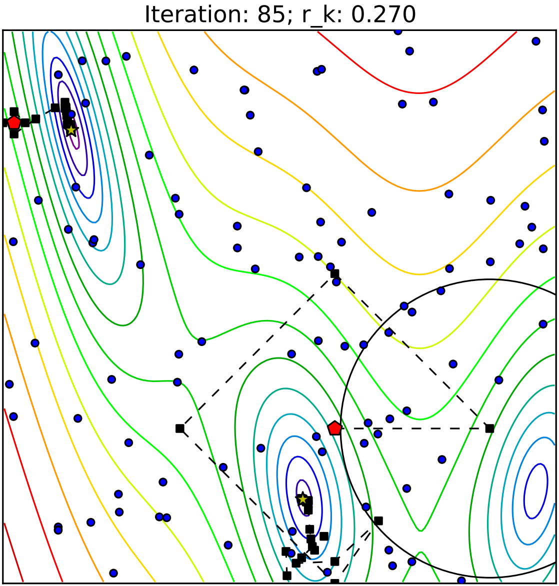
BAMLM



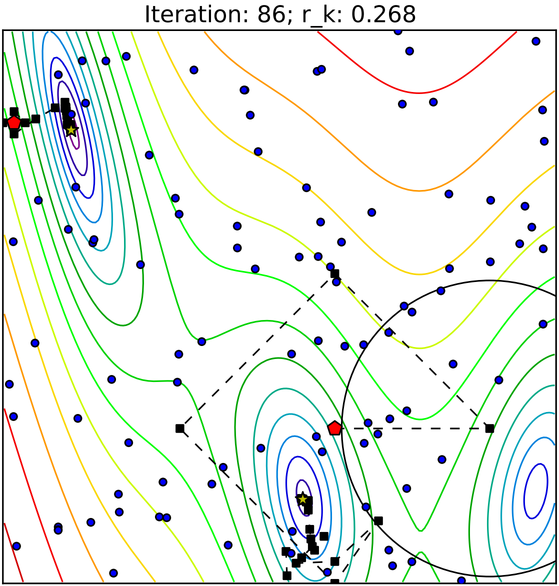
BAMLM



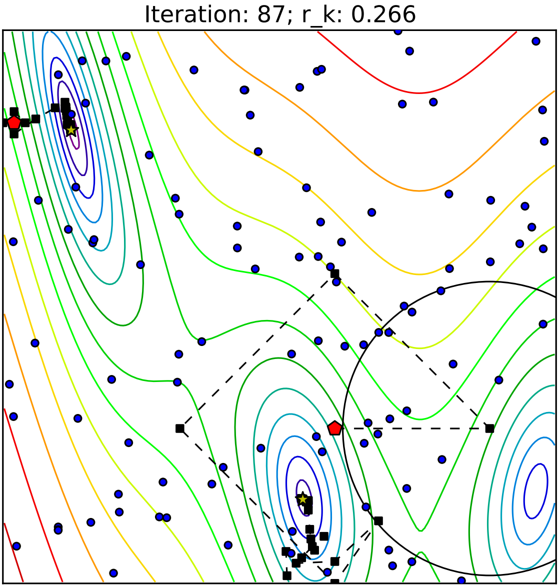
BAMLM



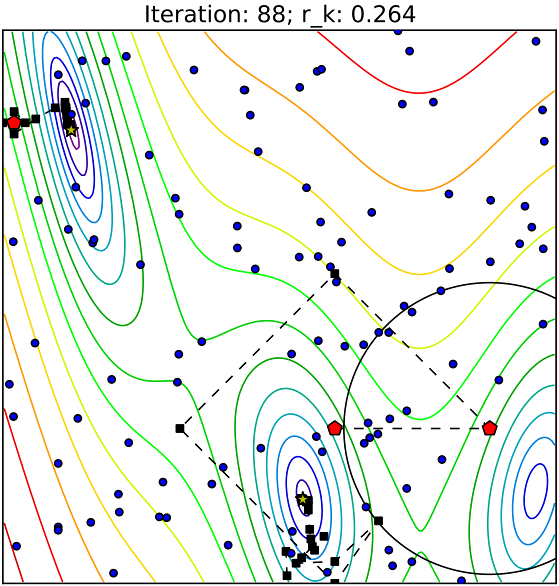
BAMLM



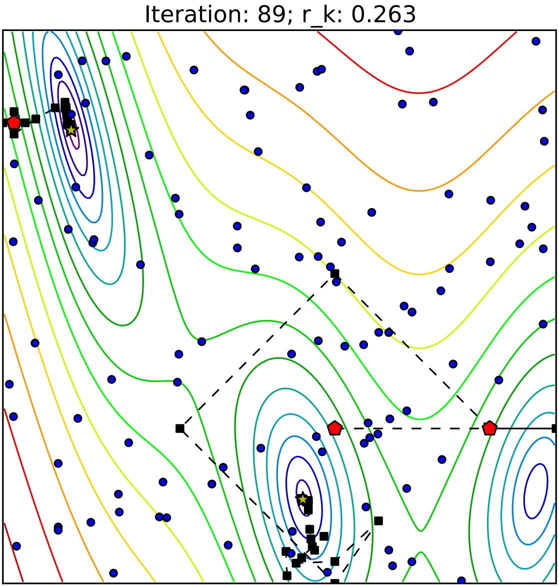
BAMLM



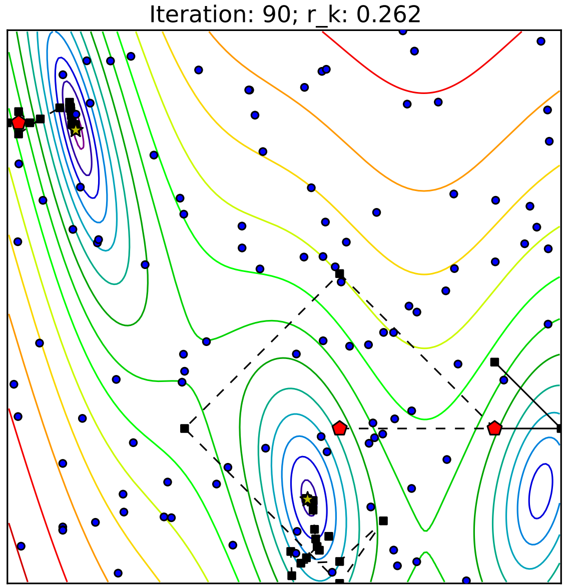
BAMLM



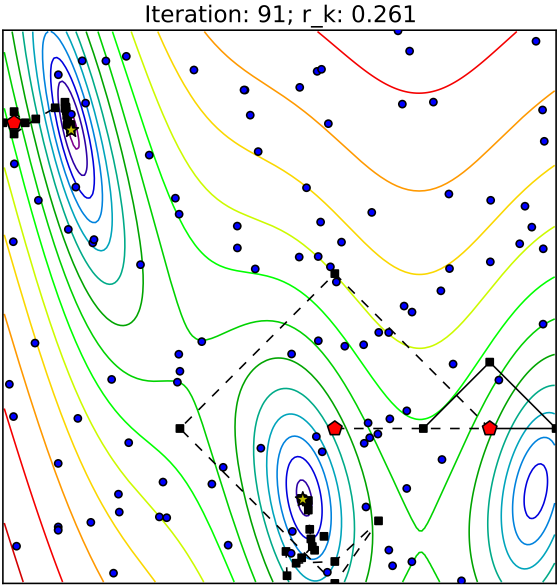
BAMLM



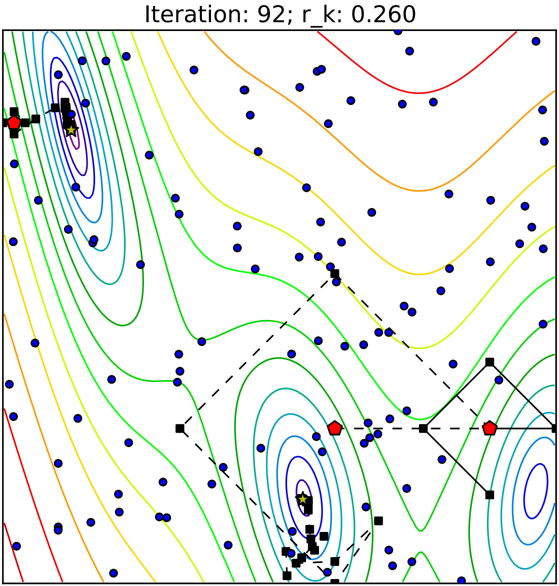
BAMLM



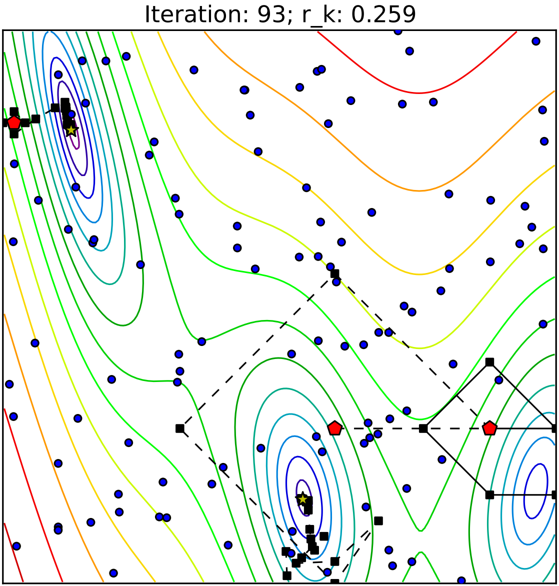
BAMLM



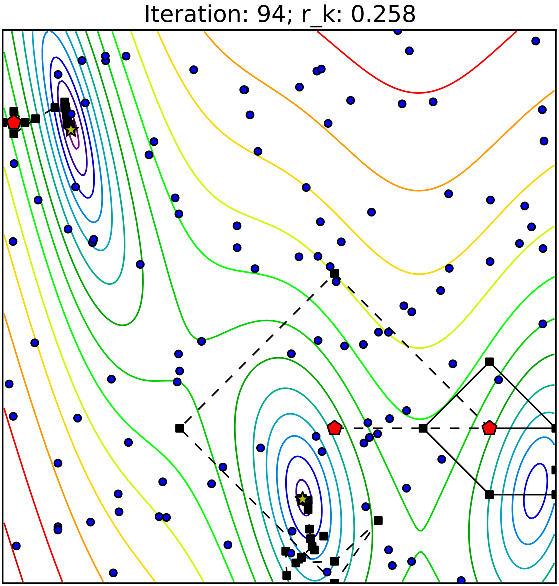
BAMLM



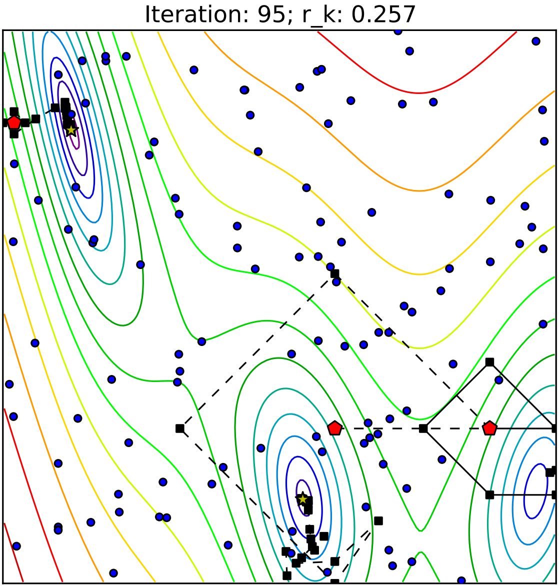
BAMLM



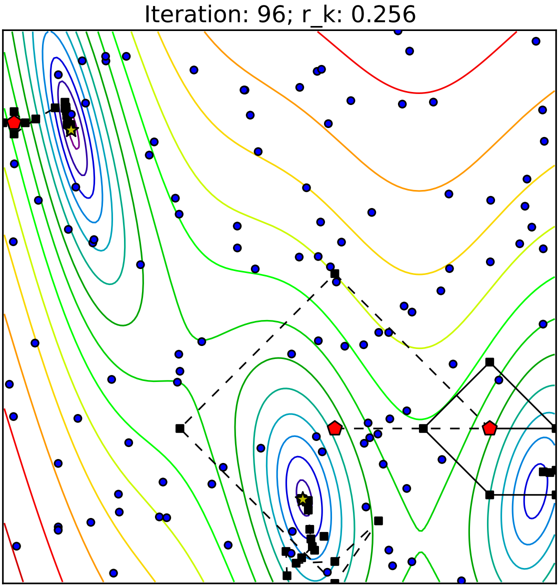
BAMLM



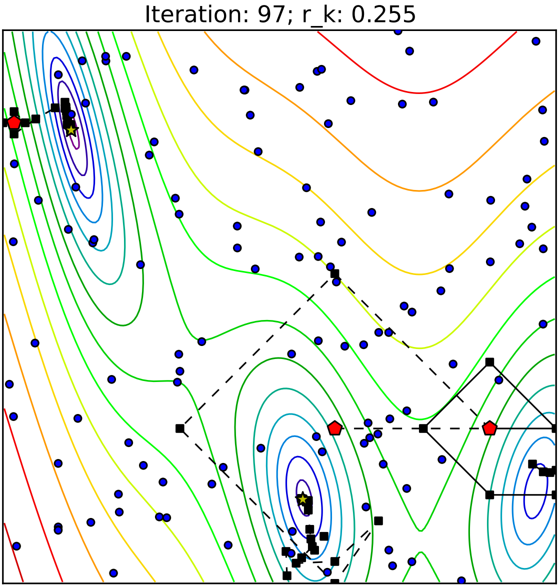
BAMLM



BAMLM

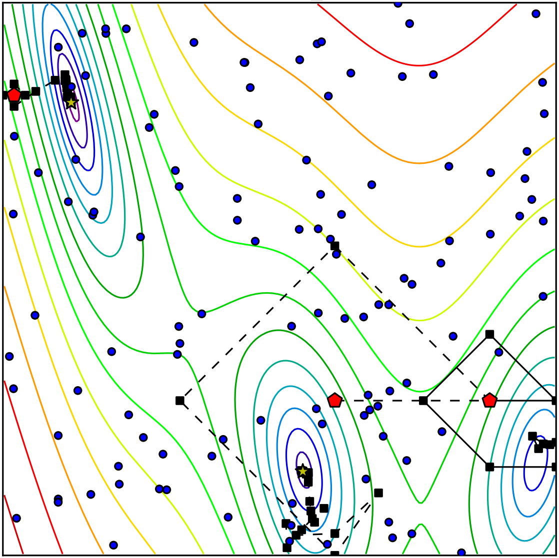


BAMLM

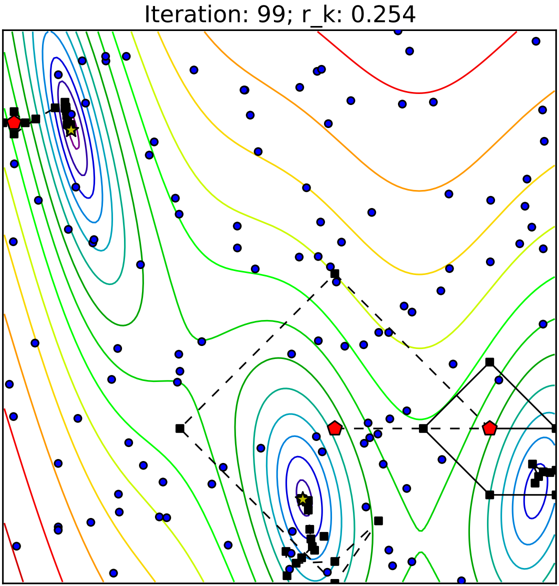


BAMLM

Iteration: 98; r_k: 0.255



BAMLM



Properties of the local optimization method

Necessary:

- Honors a starting point
- Honors bound constraints

Properties of the local optimization method

Necessary:

- Honors a starting point
- Honors bound constraints

ORBIT satisfies these [Wild, Regis, Shoemaker, SIAM-JOSC, 2008]

BOBYQA satisfies these [Powell, 2009]

Properties of the local optimization method

Necessary:

- Honors a starting point
- Honors bound constraints

ORBIT satisfies these [Wild, Regis, Shoemaker, SIAM-JOSC, 2008]

BOBYQA satisfies these [Powell, 2009]

Possibly beneficial:

- Can return multiple points of interest
- Reports solution quality/confidence at every iteration
- Can avoid certain regions in the domain
- Uses a history of past evaluations of f
- Uses additional points mid-run

AAMLM

Algorithm 3: AAMLM

Give each worker a point to evaluate
for $k = 1, 2, \dots$ **do**
  Receive from (longest waiting) worker w that has evaluated f
  Update $\mathcal{H}_k$ and r_k
  if *point evaluated by* w *is from an active run* **then**
    if *Run is complete* **then**
      Update X_k^*, and mark points inactive
    else
      Add the next point in its localopt run (not in $\mathcal{H}_k$) to Q_L
    Start run(s) at all point(s) satisfying (S1)–(S4), (L1)–(L6)
    Add the subsequent point (not in $\mathcal{H}_k$) from each run to Q_L
  Merge runs in Q_L with candidate minima within 2ν of each other
  Give w a point at which to evaluate f, either from Q_L or $\mathcal{R}$

BAMLM

MLSL: (S2)–(S4)

$\hat{x} \in \mathcal{S}_k$

(S1) $\nexists x \in \mathcal{L}_k$ with $[\|\hat{x} - x\| \leq r_k$ and $f(x) < f(\hat{x})]$

(S2) $\nexists x \in \mathcal{S}_k$ *with* $[\|\hat{x} - x\| \leq r_k$ *and* $f(x) < f(\hat{x})]$

(S3) $\hat{x}$ *has not started a local optimization run*

(S4) $\hat{x}$ *is at least* μ *from* $\partial\mathcal{D}$ *and* ν *from known local minima*

BAMLM: (S1)–(S4), (L1)–(L6)

$\hat{x} \in \mathcal{L}_k$

(L1) $\nexists x \in \mathcal{L}_k$ $[\|\hat{x} - x\| \leq r_k$ and $f(x) < f(\hat{x})]$

(L2) $\nexists x \in \mathcal{S}_k$ with $[\|\hat{x} - x\| \leq r_k$ and $f(x) < f(\hat{x})]$

(L3) $\hat{x}$ has not started a local optimization run

(L4) $\hat{x}$ is at least μ from $\partial\mathcal{D}$ and ν from known local minima

(L5) $\hat{x}$ is not in an active local optimization run and has not been ruled stationary

(L6) $\exists r_k$-descent path in $\mathcal{H}_k$ from some $x \in \mathcal{S}_k$ satisfying (S2-S4) to $\hat{x}$

AAMLM Theory

Theorem

Given the same assumptions as MLSL, AAMLM will start a finite number of local optimization runs with probability 1.

AAMLM Theory

Theorem

Given the same assumptions as MLSL, AAMLM will start a finite number of local optimization runs with probability 1.

Assumption

There exists $K_0 < \infty$ so that for any K_0 consecutive iterations, there is a positive (bounded away from zero) probability of evaluating a point from the sample stream and each existing local optimization run.

AAMLM Theory

Theorem

Given the same assumptions as MLSL, AAMLM will start a finite number of local optimization runs with probability 1.

Assumption

There exists $K_0 < \infty$ so that for any K_0 consecutive iterations, there is a positive (bounded away from zero) probability of evaluating a point from the sample stream and each existing local optimization run.

Theorem

Each $x^* \in X^*$ will almost surely be either identified in a finite number of evaluations or have a single local optimization run that is converging asymptotically to it.

Measuring Performance

GLODS Global & local optimization using direct search [Custódio, Madeira (JOGO, 2014)]

Direct Serial DIRECT [D. Finkel's MATLAB code]

pVTDirect Parallel DIRECT [He, Watson, Sosonkina (TOMS, 2009)]

Random Uniform sampling over domain (as a baseline)

BAMLM

- Concurrency: 4
- Local optimization method
  - ORBIT [Wild, Regis, & Shoemaker (SIAM JOSC, 2008)]
  - BOBYQA [Powell, 2009]
- Initial sample size: $10n$

- Each method evaluates Direct's $2n + 1$ initial points.

Measuring Performance

Notation:

Let X^* be the set of all local minima of f.
Let $f^*_{(i)}$ be the ith smallest value $\{f(x^*)|x^* \in X^*\}$.
Let $x^*_{(i)}$ be the element of X^* corresponding to the value $f^*_{(i)}$.

The global minimum has been found at a level $\tau > 0$ at batch k if an algorithm it has found a point $\hat{x}$ satisfying:

$$f(\hat{x}) - f^*_{(1)} \leq (1 - \tau)\left(f(x_0) - f^*_{(1)}\right),$$

where x_0 is the starting point for problem p.

Measuring Performance

Notation:

Let X^* be the set of all local minima of f.
Let $f^*_{(i)}$ be the ith smallest value $\{f(x^*)|x^* \in X^*\}$.
Let $x^*_{(i)}$ be the element of X^* corresponding to the value $f^*_{(i)}$.

The j best local minima have been found at a level $\tau > 0$ at batch k if:

$$\left|\left\{x^*_{(1)}, \ldots, x^*_{(\underline{j}-1)}\right\} \bigcap \left\{x^*_{(i)} : \exists x \in \mathcal{H}_k \text{ with } \left\|x - x^*_{(i)}\right\| \le r_n(\tau)\right\}\right| = \underline{j} - 1$$

$$\&$$

$$\left|\left\{x^*_{(\underline{j})}, \ldots, x^*_{(\bar{j})}\right\} \bigcap \left\{x^*_{(i)} : \exists x \in \mathcal{H}_k \text{ with } \left\|x - x^*_{(i)}\right\| \le r_n(\tau)\right\}\right| \ge j - \underline{j} + 1,$$

where $\underline{j}$ and $\bar{j}$ are the smallest and largest integers such that $f^*_{(\bar{j})} = f^*_{(j)} = f^*_{(\underline{j})}$ and where $r_n(\tau) = \sqrt[n]{\frac{\tau \operatorname{vol}(\mathcal{D})\Gamma(\frac{n}{2}+1)}{\pi^{n/2}}}$.

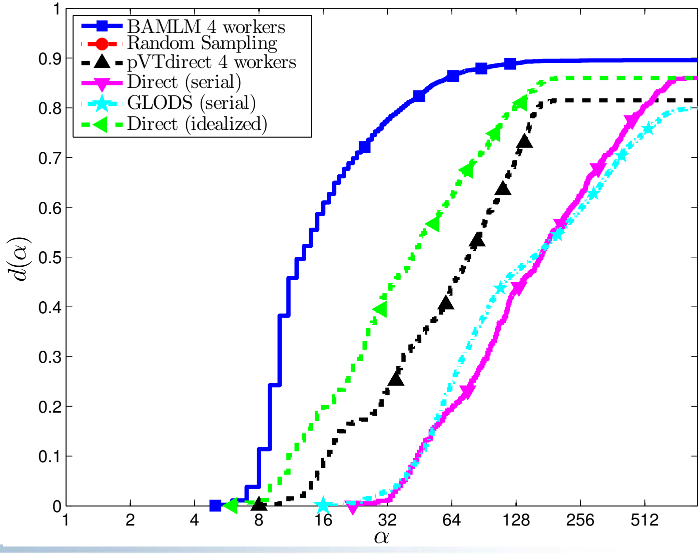
Problems considered

GKLS problem generator [Gaviano et al., "Algorithm 829" (TOMS, 2003)]

- 600 synthetic problems with known local minima
- $n = 2, \ldots, 7$
- 10 local minima in the unit cube with a unique global minimum
- 100 problems for each dimension
- 5 replications (different seeds) for each problem
- 5000 evaluations

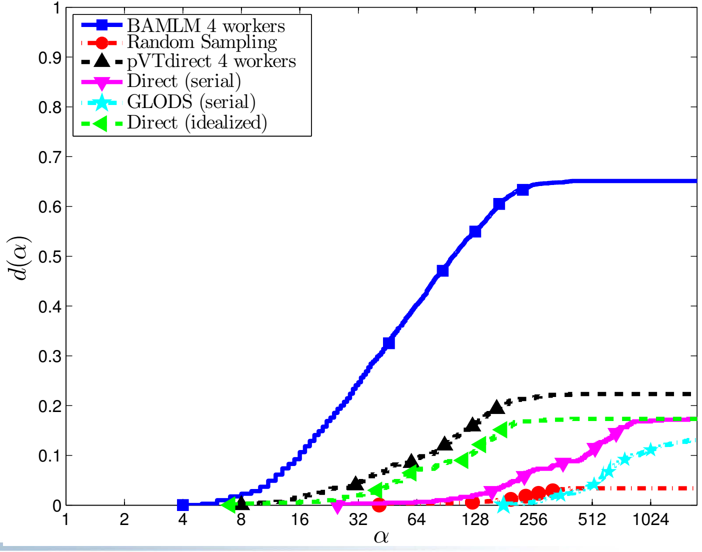
Data Profiles

$$f(x) - f^*_{(1)} \leq (1 - 10^{-5}) \left(f(x_0) - f^*_{(1)} \right)$$



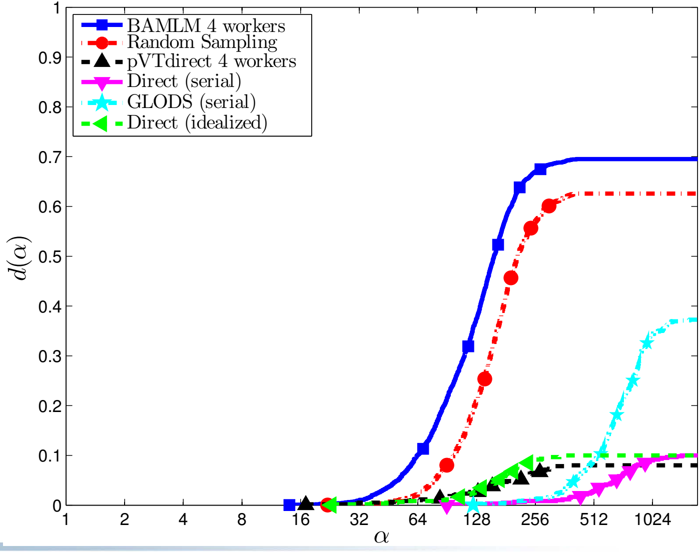
Data Profiles

Within $\sqrt[n]{\frac{10^{-4}\Gamma(\frac{n}{2}+1)}{\pi^{n/2}}}$ of 3 best minima



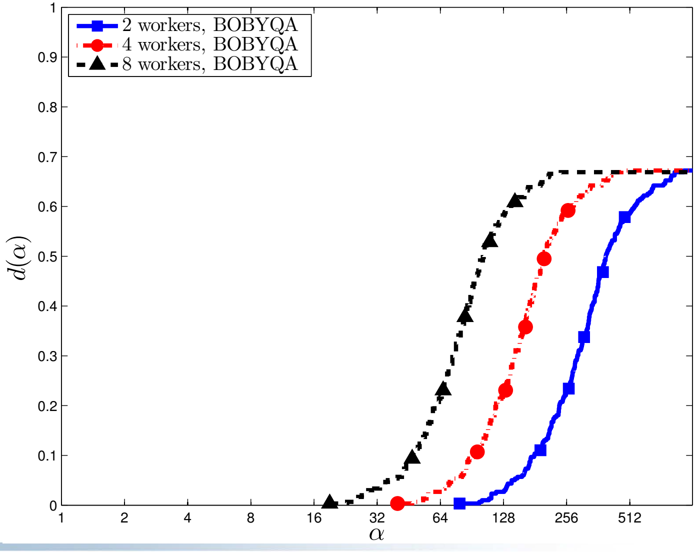
Data Profiles

Within $\sqrt[n]{\frac{10^{-3}\Gamma(\frac{n}{2}+1)}{\pi^{n/2}}}$ of 7 best minima



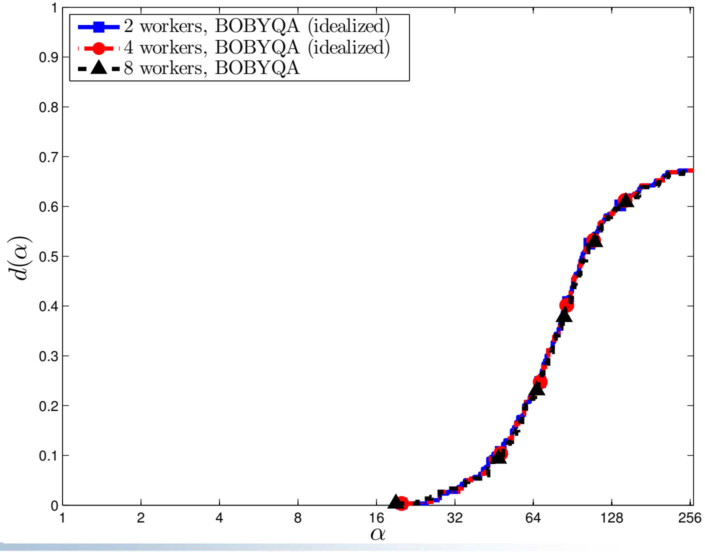
Data Profiles

Within $\sqrt[n]{\frac{10^{-3}\Gamma(\frac{n}{2}+1)}{\pi^{n/2}}}$ of 7 best minima



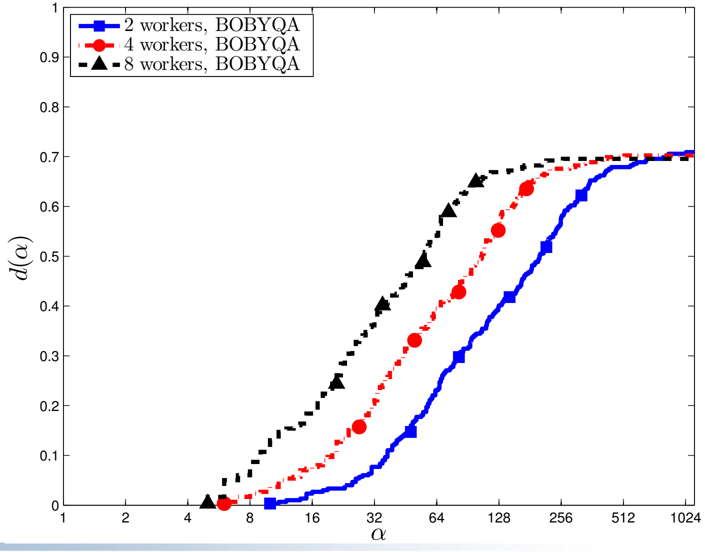
Data Profiles

Within $\sqrt[n]{\frac{10^{-3}\Gamma(\frac{n}{2}+1)}{\pi^{n/2}}}$ of 7 best minima



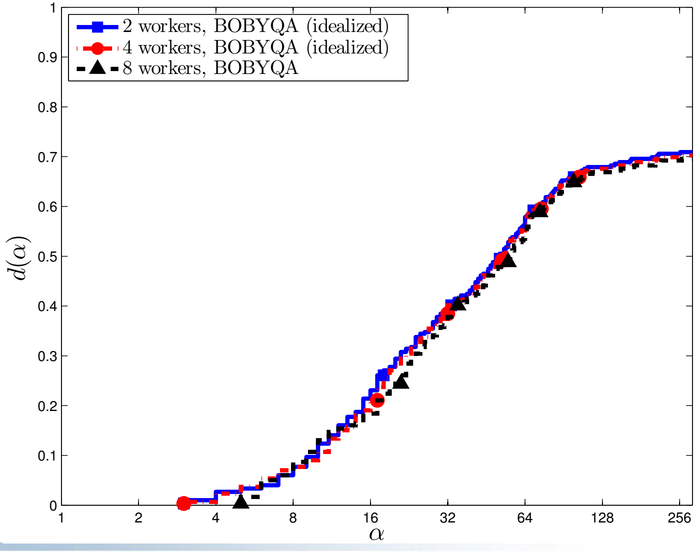
Data Profiles

Within $\sqrt[n]{\frac{10^{-4}\Gamma(\frac{n}{2}+1)}{\pi^{n/2}}}$ of 3 best minima

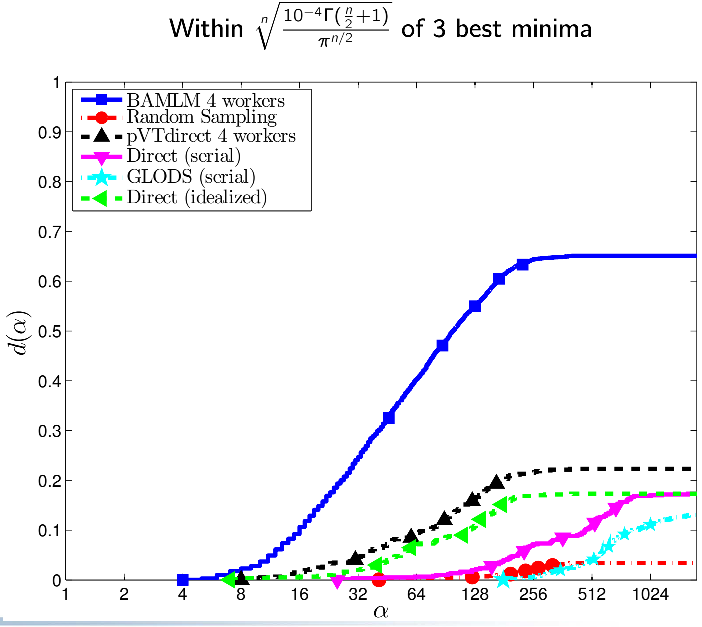


Data Profiles

Within $\sqrt[n]{\frac{10^{-4}\Gamma(\frac{n}{2}+1)}{\pi^{n/2}}}$ of 3 best minima

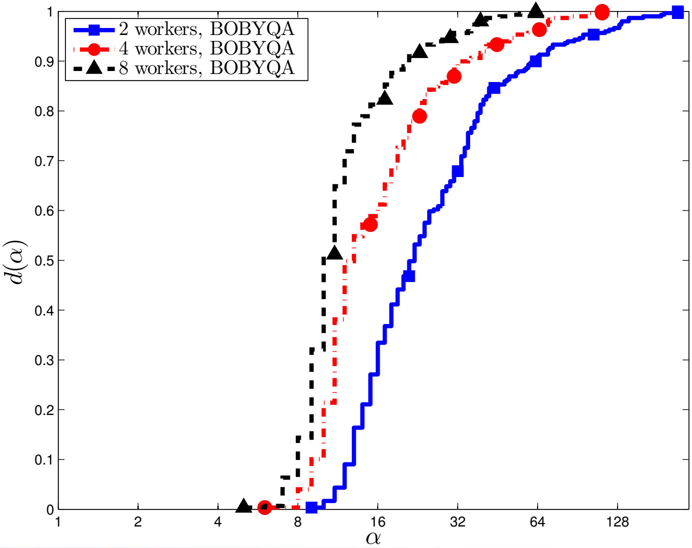


Data Profiles



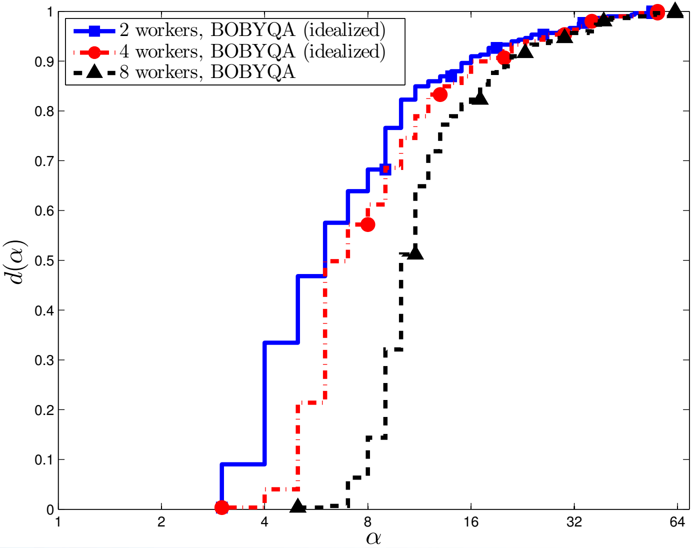
Data Profiles

$$f(x) - f^*_{(1)} \leq (1 - 10^{-5}) \left(f(x_0) - f^*_{(1)} \right)$$

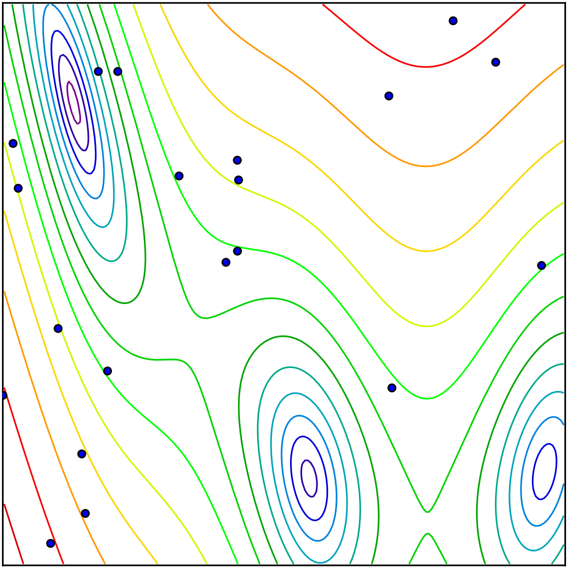


Data Profiles

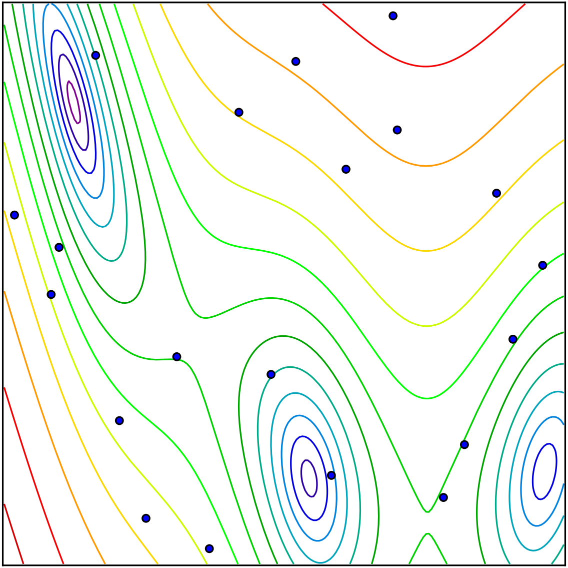
$$f(x) - f^*_{(1)} \leq (1 - 10^{-5}) \left(f(x_0) - f^*_{(1)} \right)$$



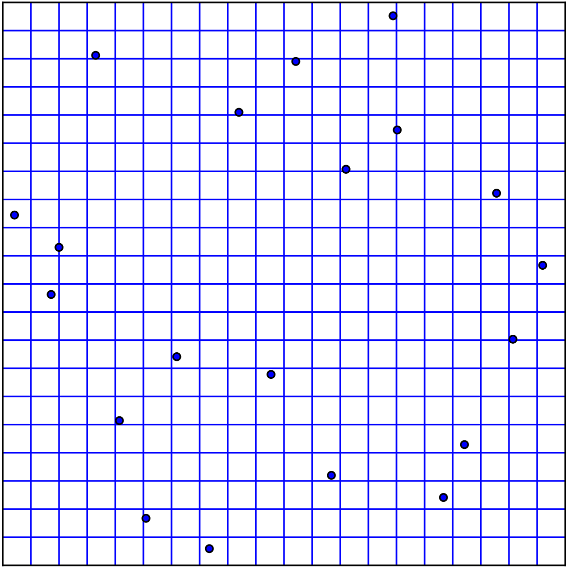
Uniform sampling



Latin hypercube sampling



Latin hypercube sampling



BAMLM with LHS

Critical distance for uniform sampling:

$$r_k = \pi^{-1/2} \left(\Gamma(1 + \frac{n}{2}) \, \mathrm{vol}\,(\mathcal{D}) \, \frac{\sigma \log kN}{kN} \right)^{1/n}$$

Critical distance for Latin hypercube sampling:

$$r_k = \pi^{-1/2} \left(\Gamma(1 + \frac{n}{2}) \, \mathrm{vol}\,(\mathcal{D}) \, \frac{\sigma N^{n-1} \log k}{k} \right)^{1/n} \tag{2}$$

BAMLM with LHS

Critical distance for uniform sampling:

$$r_k = \pi^{-1/2} \left(\Gamma(1 + \frac{n}{2}) \operatorname{vol}(\mathcal{D}) \frac{\sigma \log kN}{kN} \right)^{1/n}$$

Critical distance for Latin hypercube sampling:

$$r_k = \pi^{-1/2} \left(\Gamma(1 + \frac{n}{2}) \operatorname{vol}(\mathcal{D}) \frac{\sigma N^{n-1} \log k}{k} \right)^{1/n} \tag{2}$$

Theorem

If r_k is defined by (2) with $\sigma > 4$, even if the sampling continues forever, the total number of local runs started by BAMLM (or AAMLM) is finite almost surely.

Does LHS help?

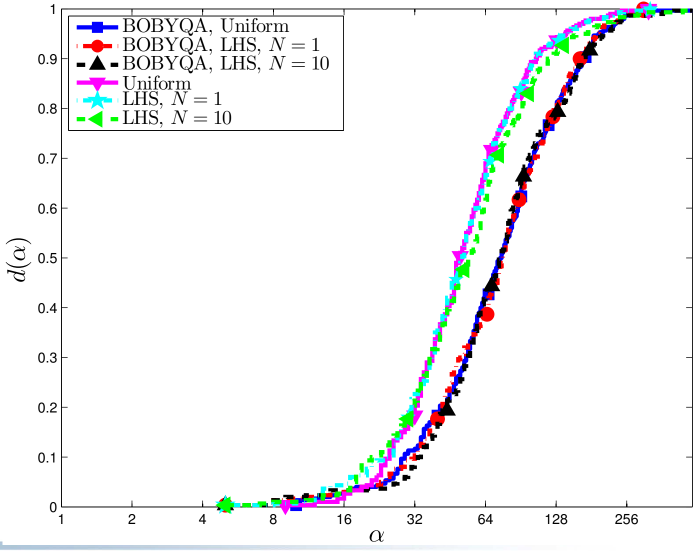
Problem setup:

- 10 different GKLS problems
- 5 different seeds
- $n = 2, \ldots, 7$

- Same starting LHS sample of $10n$ points (except for uniform)
- Same (uniform) r_k value

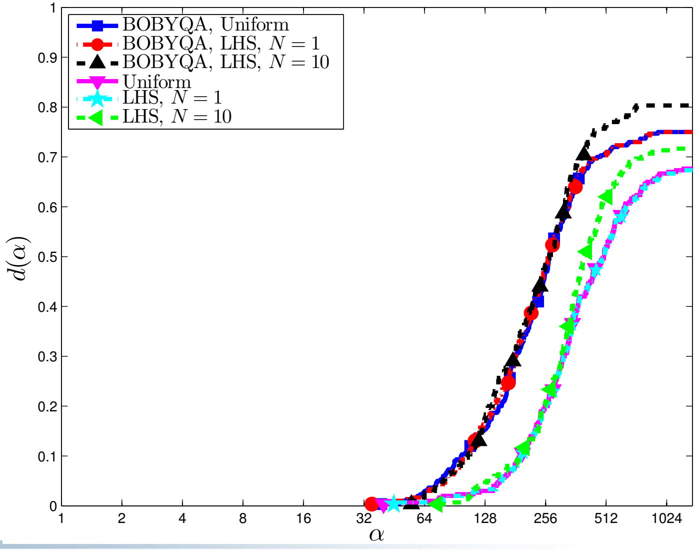
Data Profiles

Within $\sqrt[n]{\frac{10^{-2}\Gamma(\frac{n}{2}+1)}{\pi^{n/2}}}$ of 5 best minima



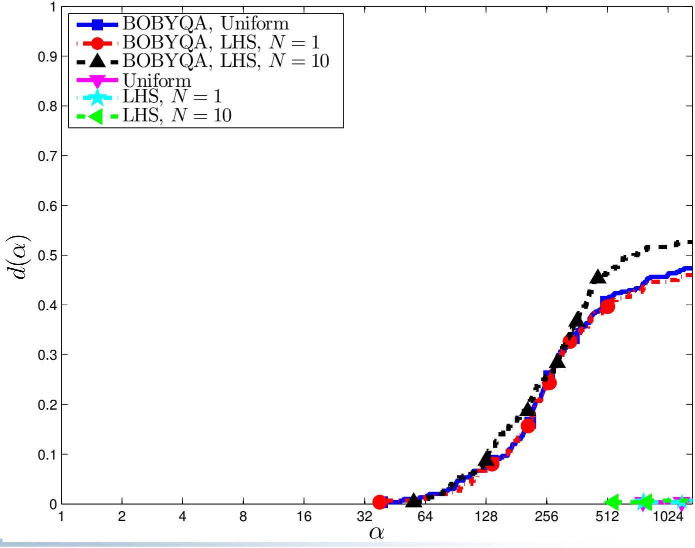
Data Profiles

Within $\sqrt[n]{\frac{10^{-3}\Gamma(\frac{n}{2}+1)}{\pi^{n/2}}}$ of 5 best minima



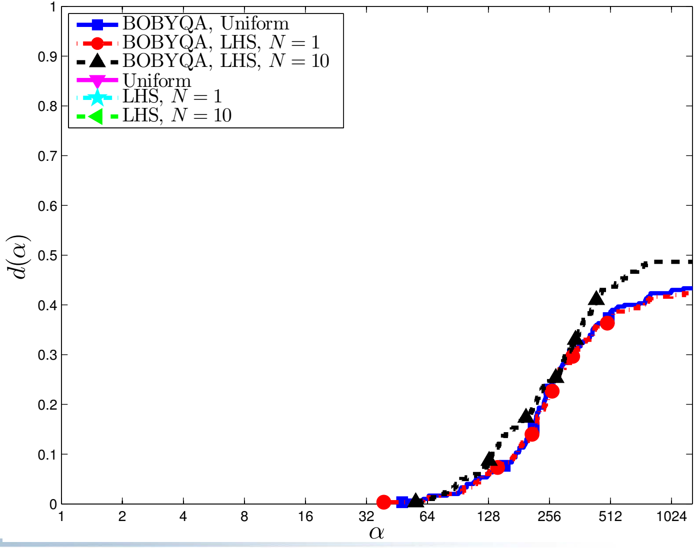
Data Profiles

Within $\sqrt[n]{\frac{10^{-4}\Gamma(\frac{n}{2}+1)}{\pi^{n/2}}}$ of 5 best minima



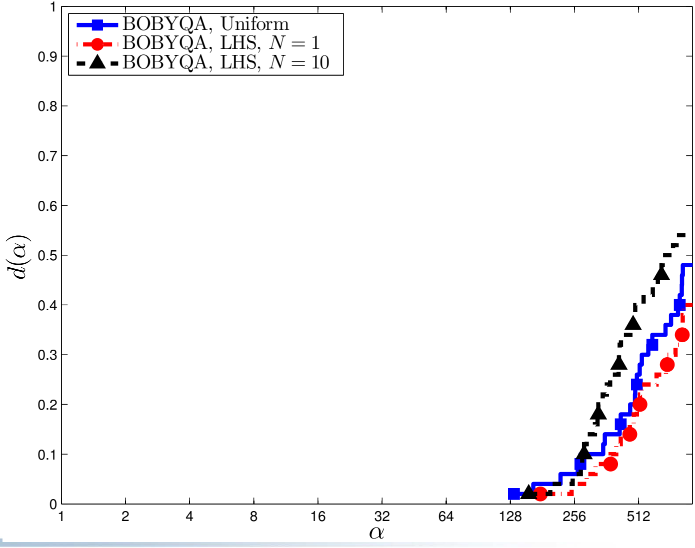
Data Profiles

Within $\sqrt[n]{\frac{10^{-5}\Gamma(\frac{n}{2}+1)}{\pi^{n/2}}}$ of 5 best minima



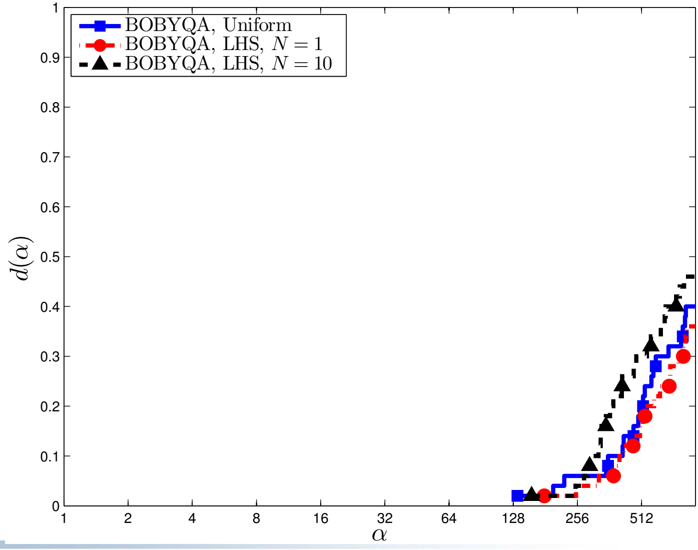
Data Profiles

Within $\sqrt[n]{\frac{10^{-4}\Gamma(\frac{n}{2}+1)}{\pi^{n/2}}}$ of 5 best minima, $n = 6$



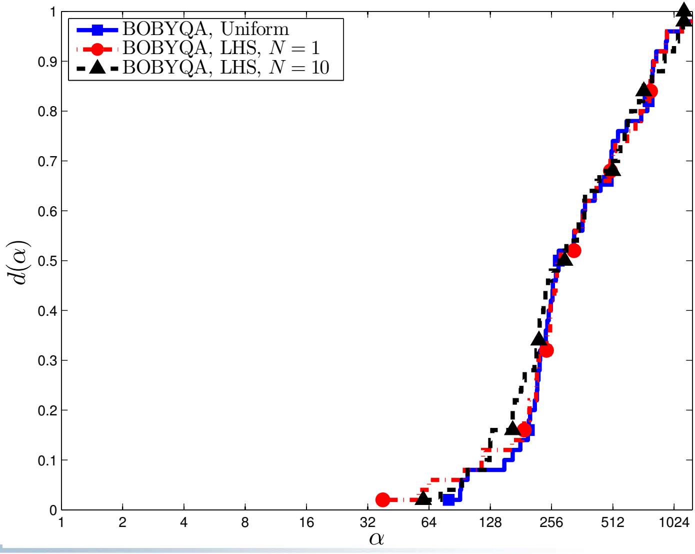
Data Profiles

Within $\sqrt[n]{\frac{10^{-5}\Gamma(\frac{n}{2}+1)}{\pi^{n/2}}}$ of 5 best minima, $n = 6$



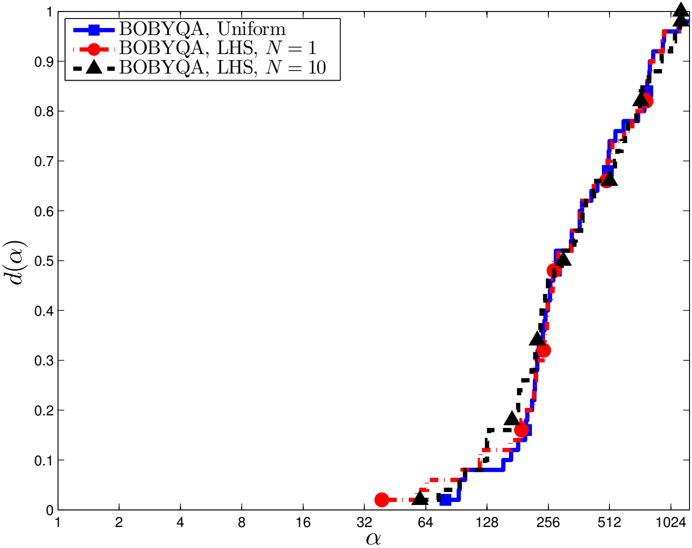
Data Profiles

Within $\sqrt[n]{\frac{10^{-4}\Gamma(\frac{n}{2}+1)}{\pi^{n/2}}}$ of 5 best minima, $n = 3$



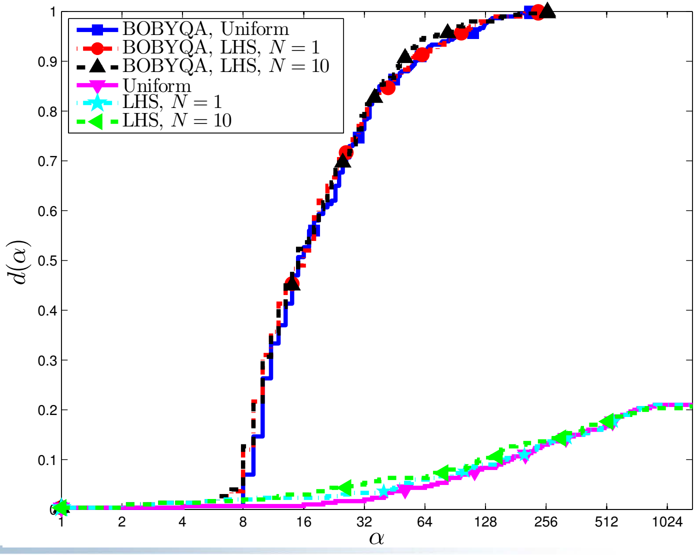
Data Profiles

Within $\sqrt[n]{\frac{10^{-5}\Gamma(\frac{n}{2}+1)}{\pi^{n/2}}}$ of 5 best minima, $n = 3$



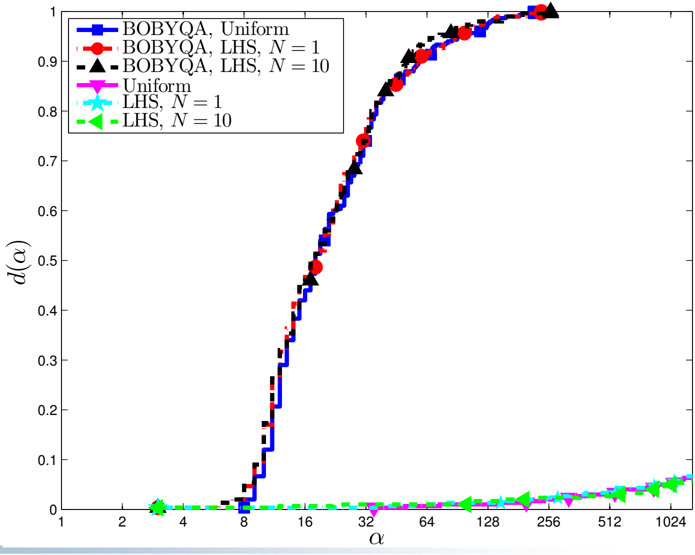
Data Profiles

$$f(x) - f^*_{(1)} \leq (1 - 10^{-2}) \left(f(x_0) - f^*_{(1)} \right)$$



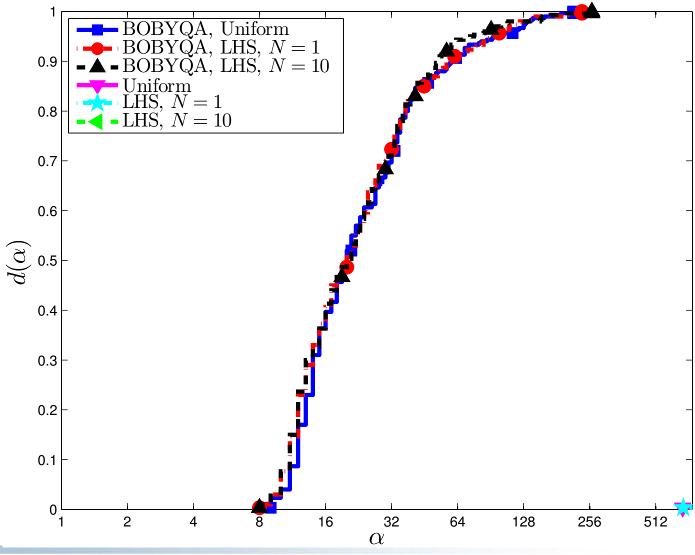
Data Profiles

$$f(x) - f^*_{(1)} \leq (1 - 10^{-3}) \left(f(x_0) - f^*_{(1)} \right)$$



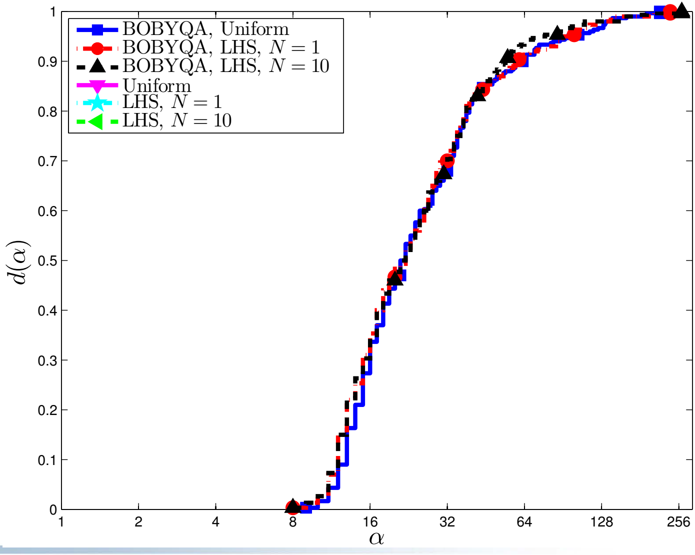
Data Profiles

$$f(x) - f^*_{(1)} \leq (1 - 10^{-4})\left(f(x_0) - f^*_{(1)}\right)$$



Data Profiles

$$f(x) - f^*_{(1)} \le (1 - 10^{-5})\left(f(x_0) - f^*_{(1)}\right)$$



Closing Remarks

- Concurrent function evaluations can locate multiple minima while efficiently finding a global minimum.

Closing Remarks

- Concurrent function evaluations can locate multiple minima while efficiently finding a global minimum.
- Latin hypercube sampling appears to help find more minima in higher-dimensional problems.

Questions:

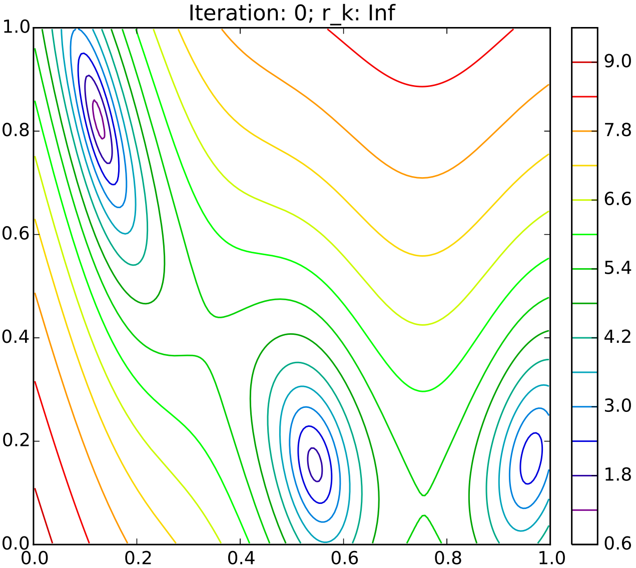
- Finding (or designing) the best local solver for our framework?
- Best way to process the queue?

AAMLM

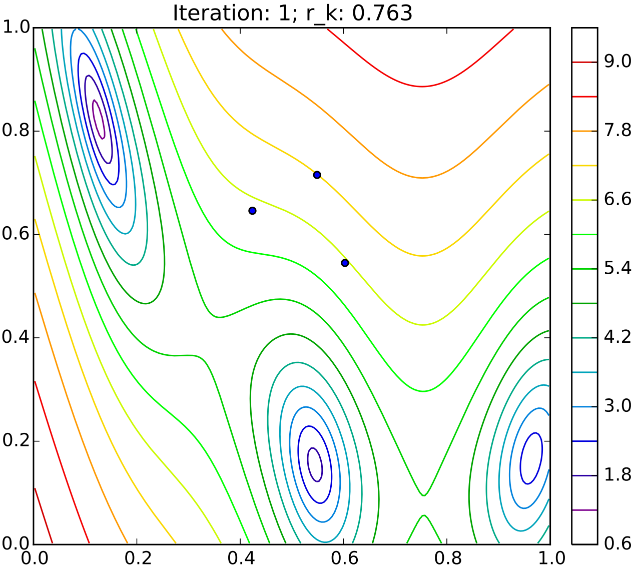
Algorithm 3: AAMLM

Give each worker a point to evaluate
for $k = 1, 2, \ldots$ **do**
  Receive from (longest waiting) worker w that has evaluated f
  Update $\mathcal{H}_k$ and r_k
  if *point evaluated by* w *is from an active run* **then**
    if *Run is complete* **then**
      Update X_k^*, and mark points inactive
    else
      Add the next point in its localopt run (not in $\mathcal{H}_k$) to Q_L
    Start run(s) at all point(s) satisfying (S1)–(S4), (L1)–(L6)
    Add the subsequent point (not in $\mathcal{H}_k$) from each run to Q_L
  Merge runs in Q_L with candidate minima within 2ν of each other
  Give w a point at which to evaluate f, either from Q_L or $\mathcal{R}$

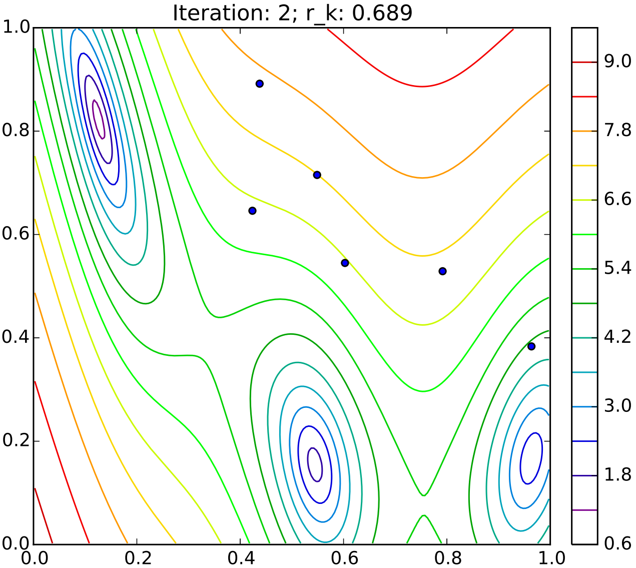
Pausing runs



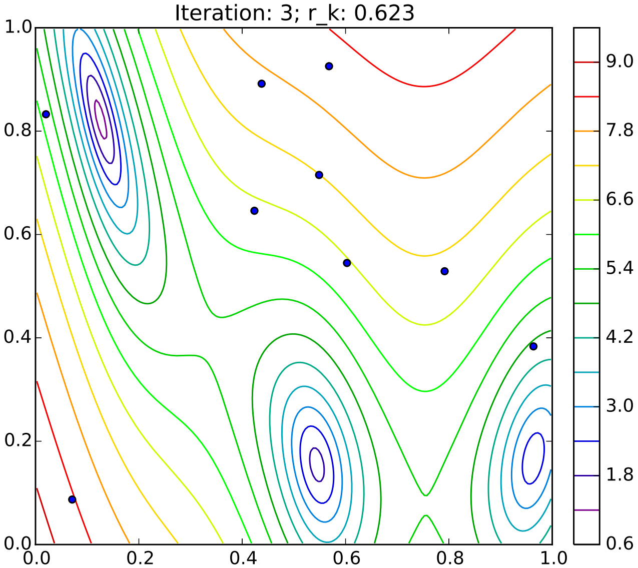
Pausing runs



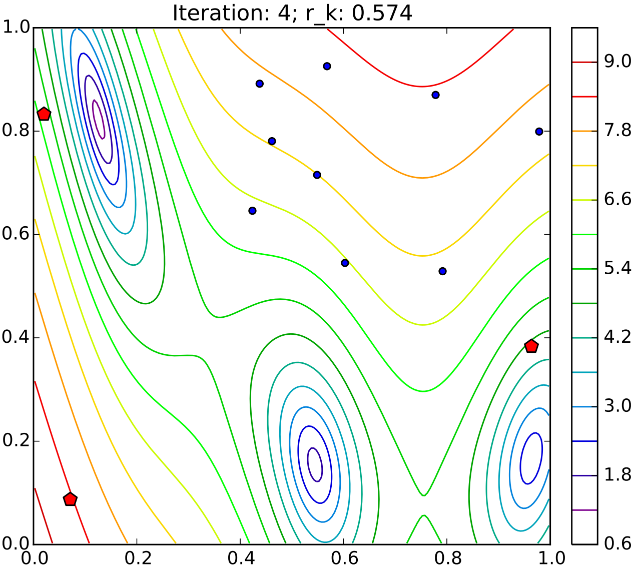
Pausing runs



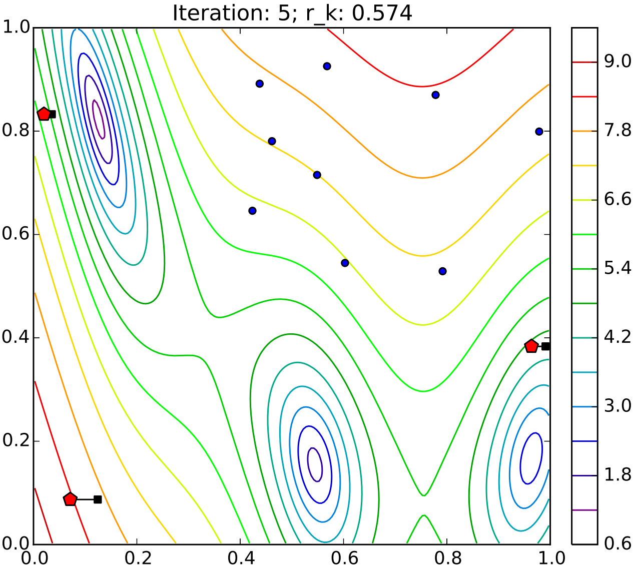
Pausing runs



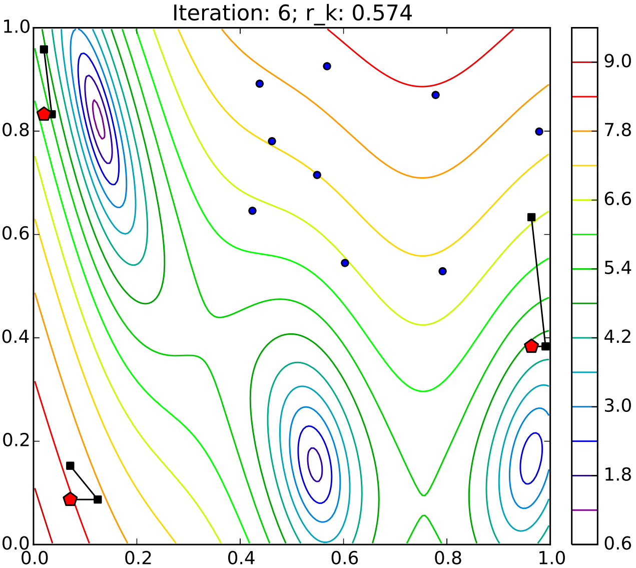
Pausing runs



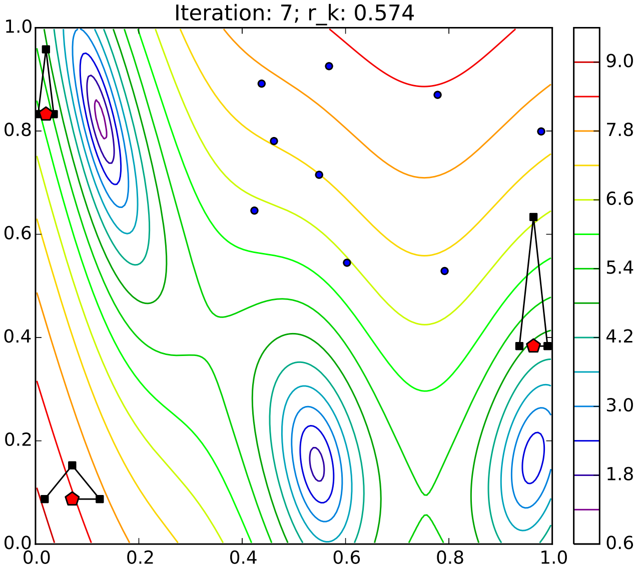
Pausing runs



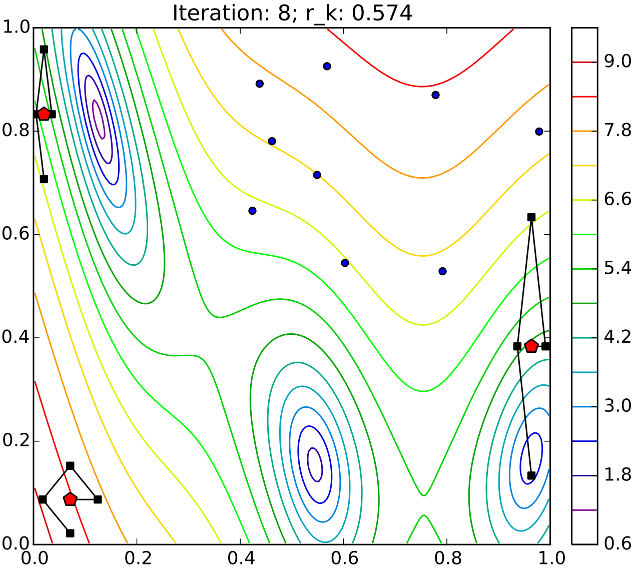
Pausing runs



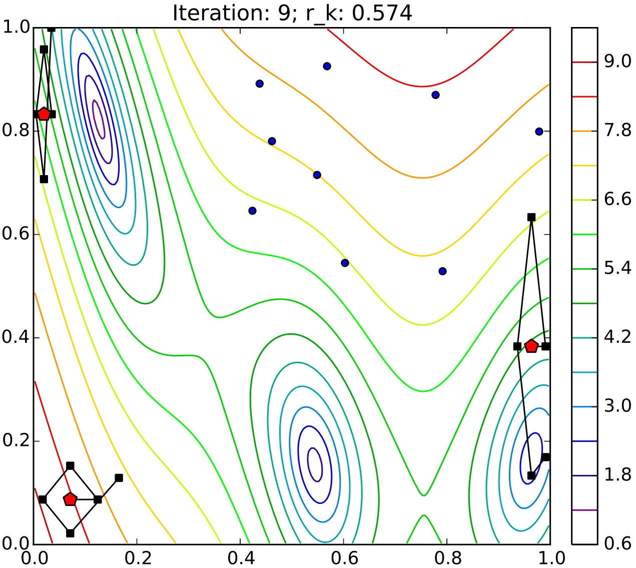
Pausing runs



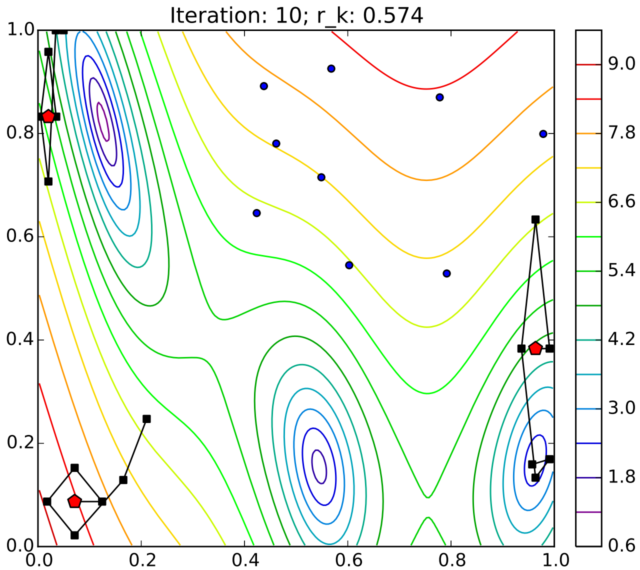
Pausing runs



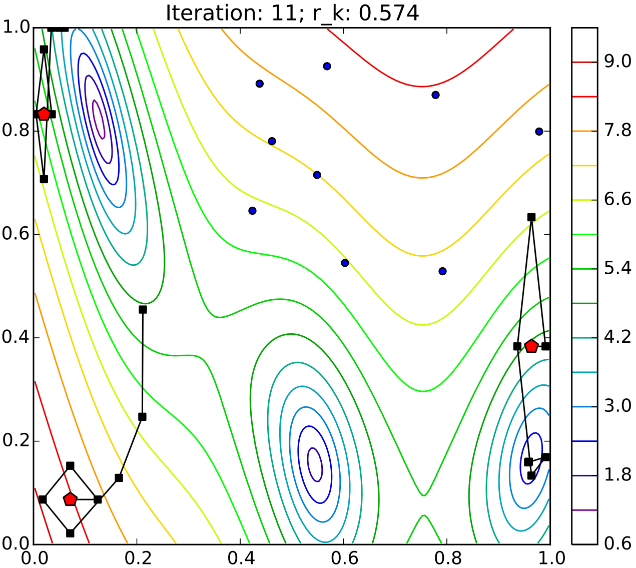
Pausing runs



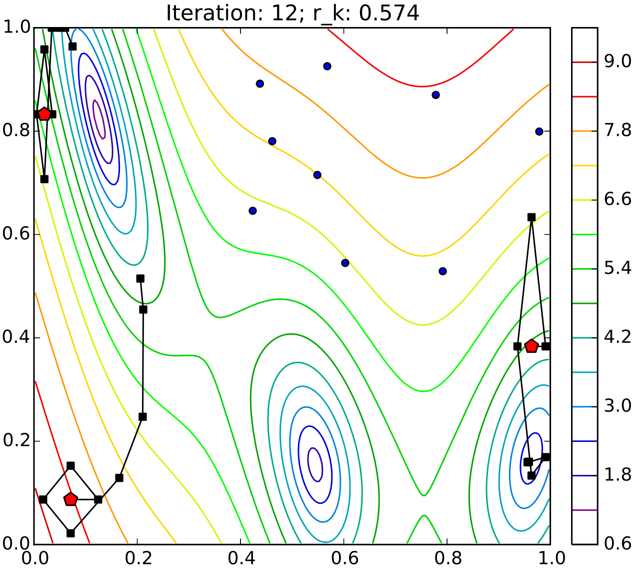
Pausing runs



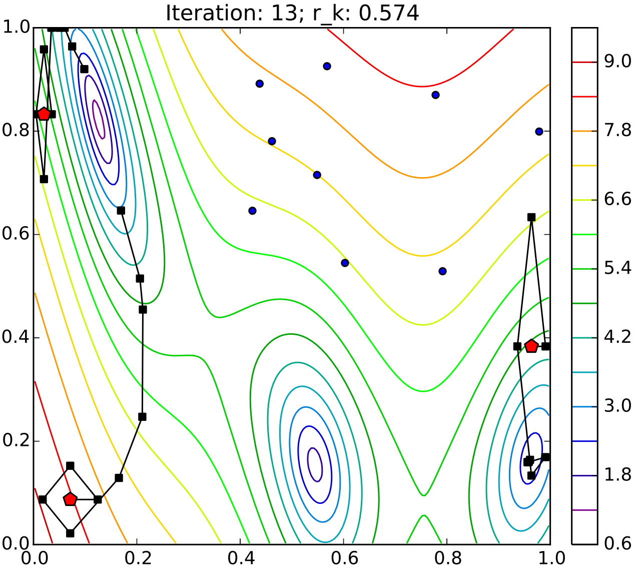
Pausing runs



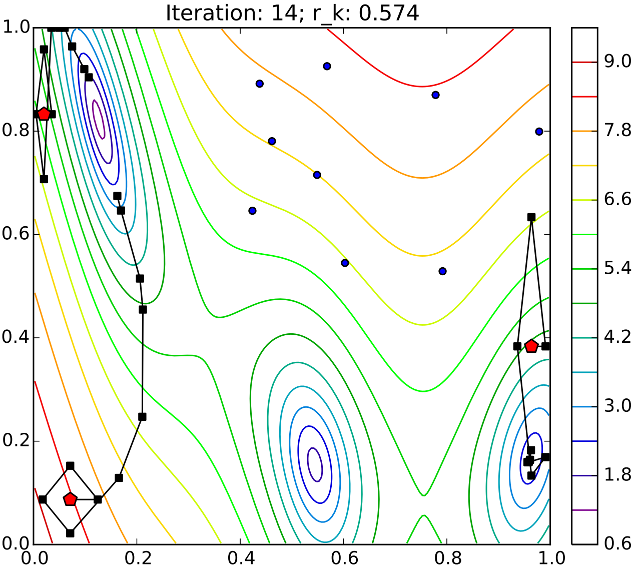
Pausing runs



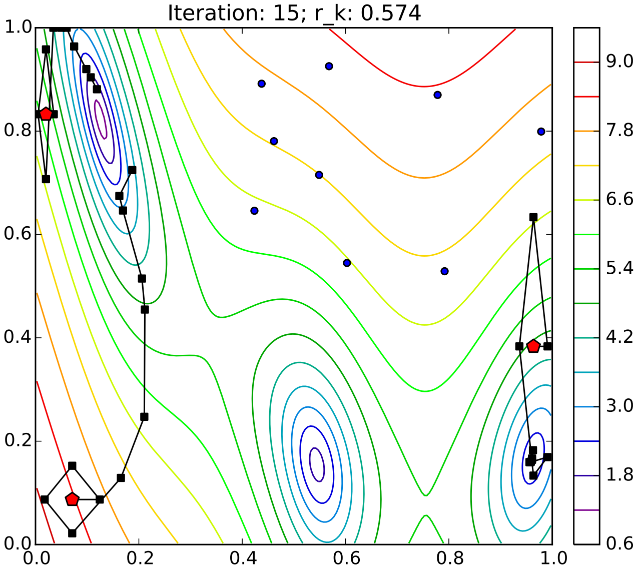
Pausing runs



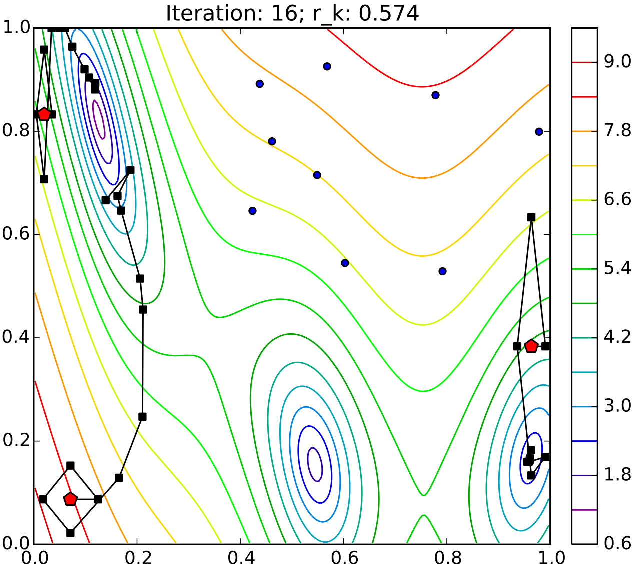
Pausing runs



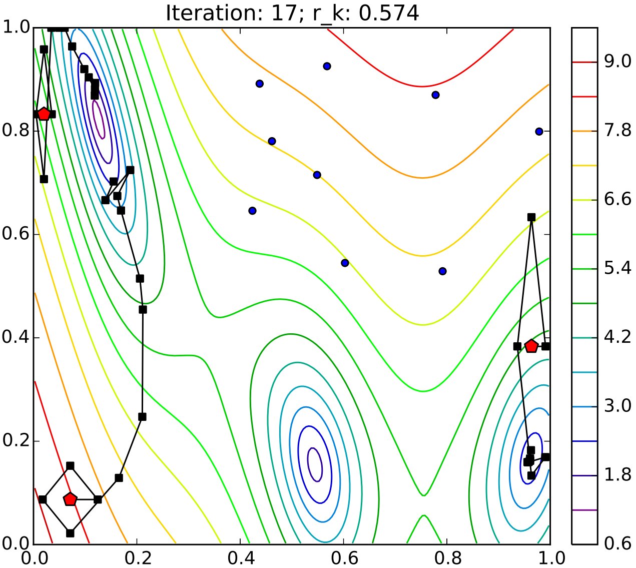
Pausing runs



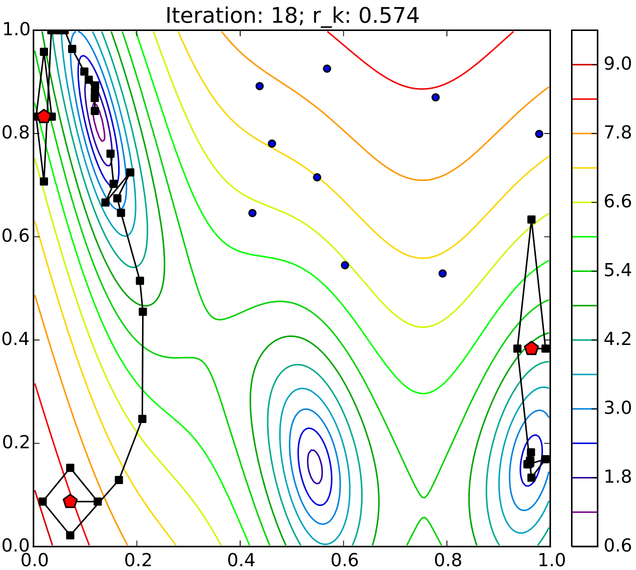
Pausing runs



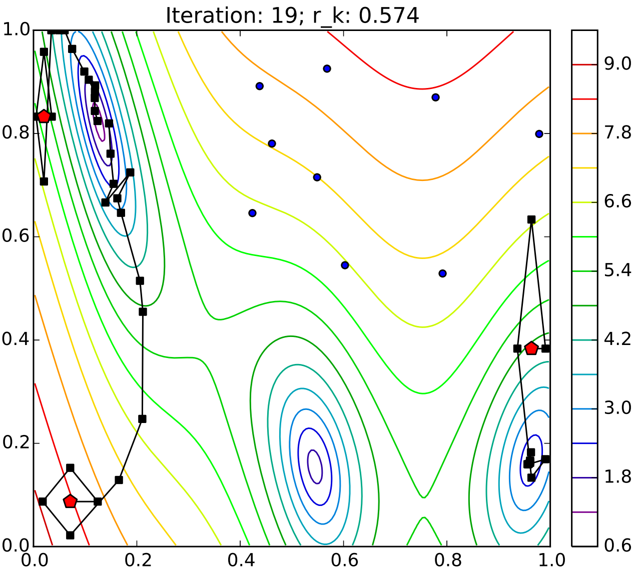
Pausing runs



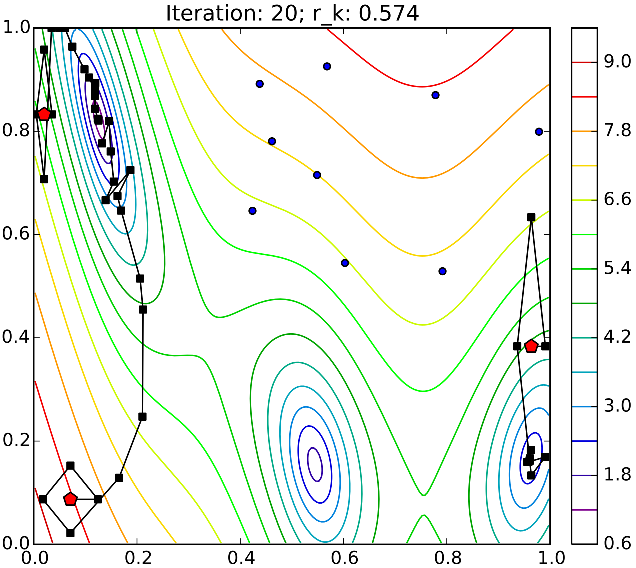
Pausing runs



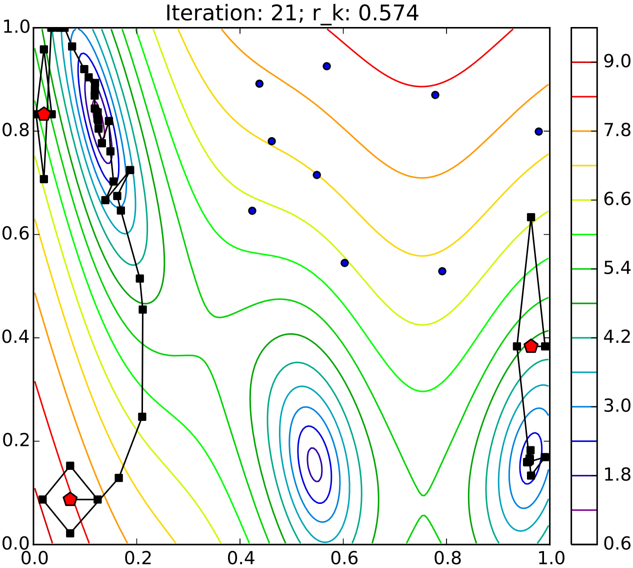
Pausing runs



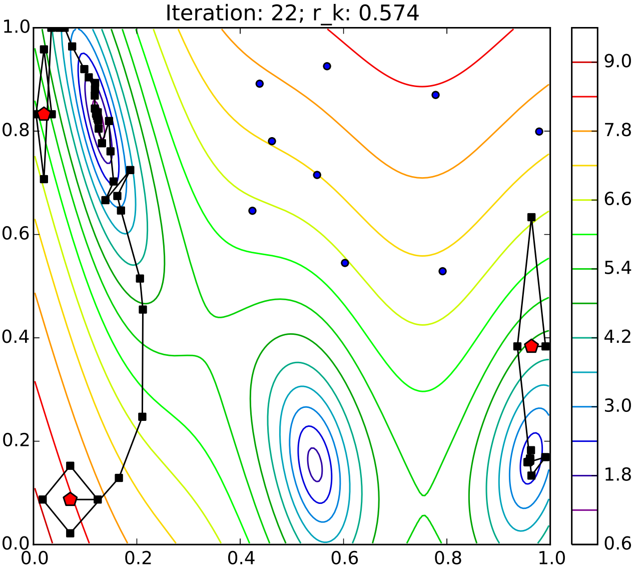
Pausing runs



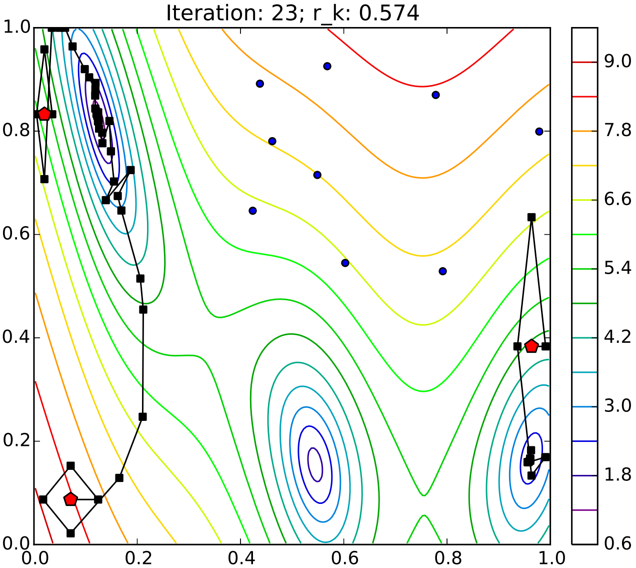
Pausing runs



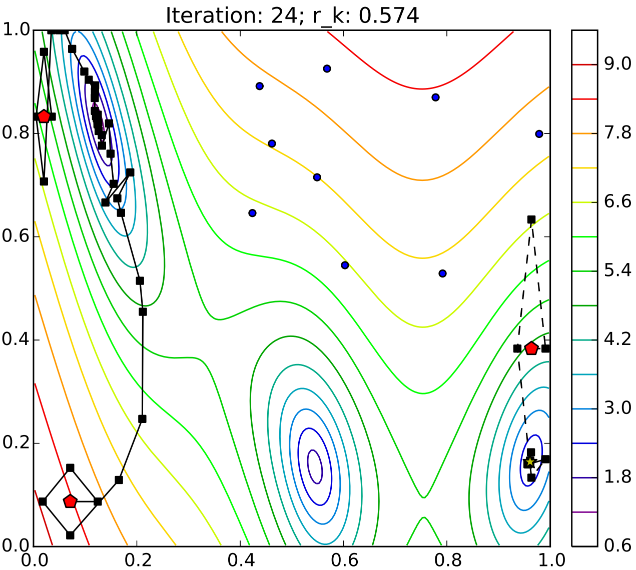
Pausing runs



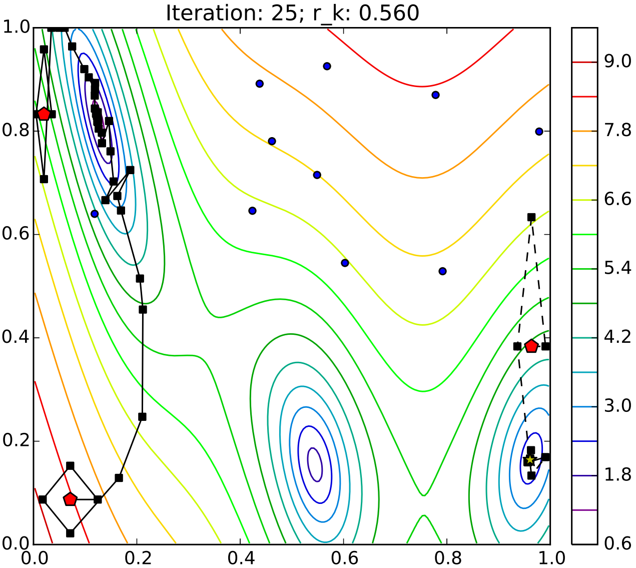
Pausing runs



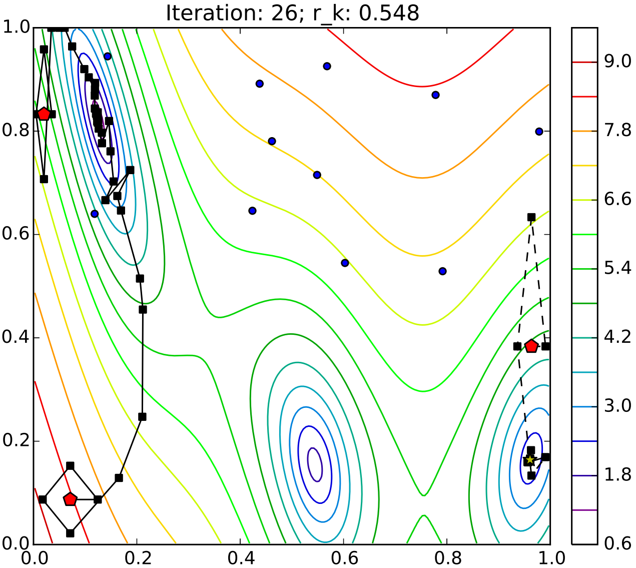
Pausing runs



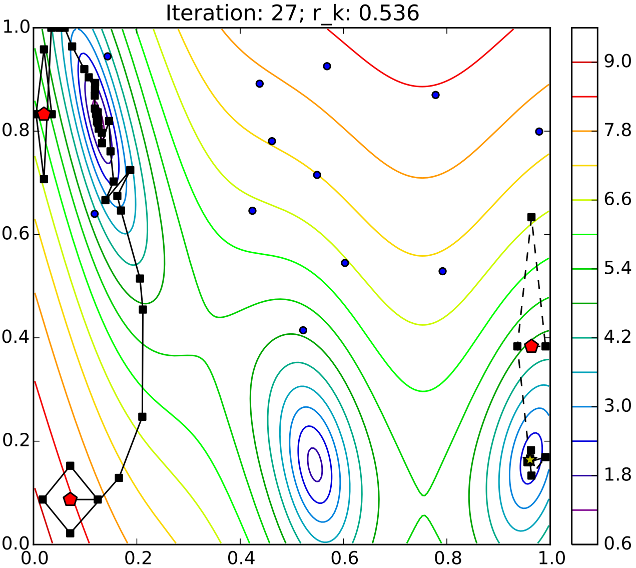
Pausing runs



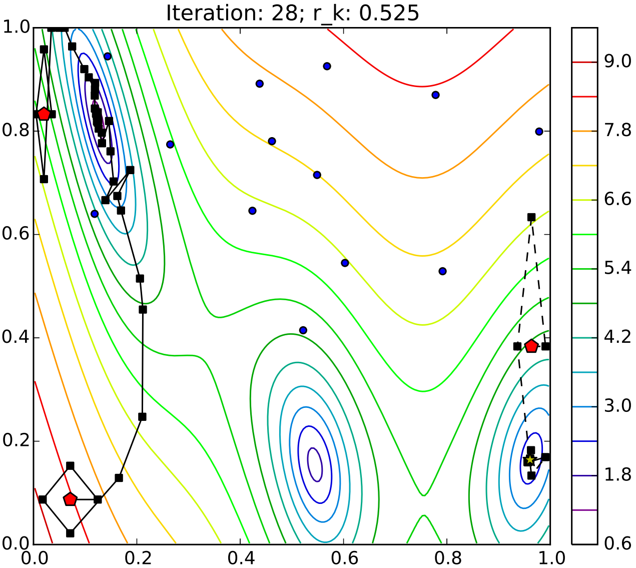
Pausing runs



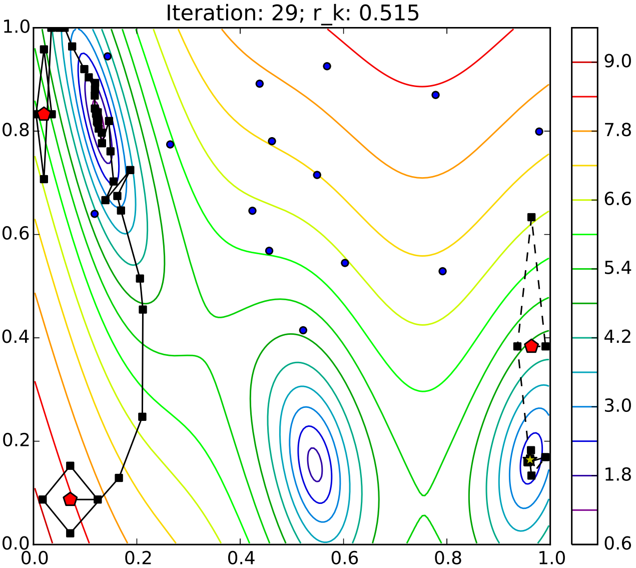
Pausing runs



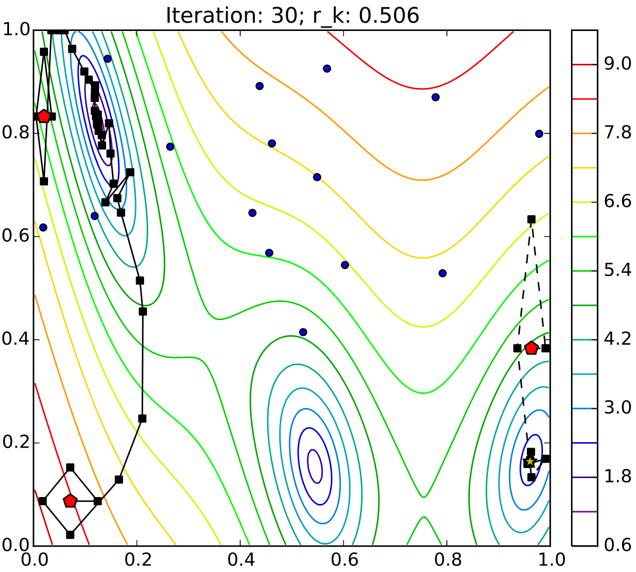
Pausing runs



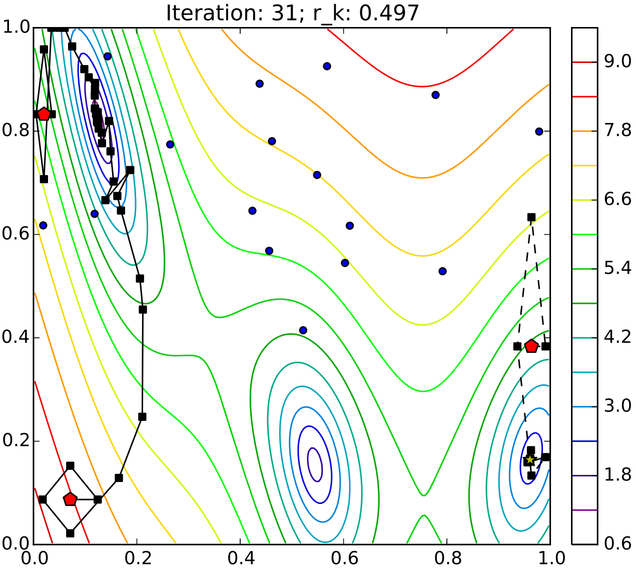
Pausing runs



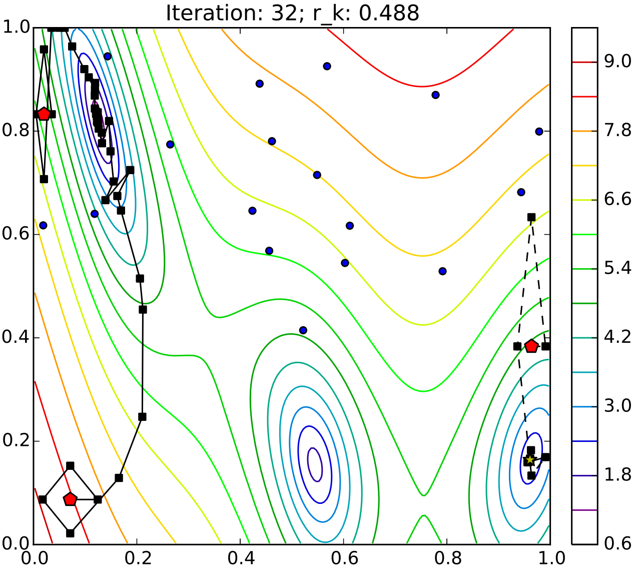
Pausing runs



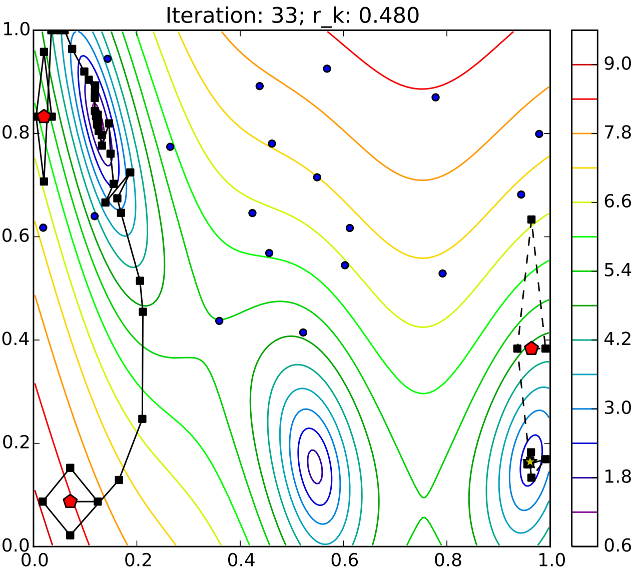
Pausing runs



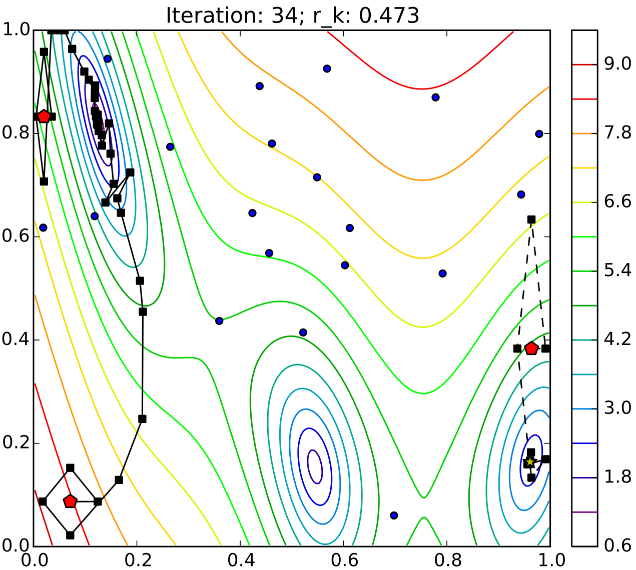
Pausing runs



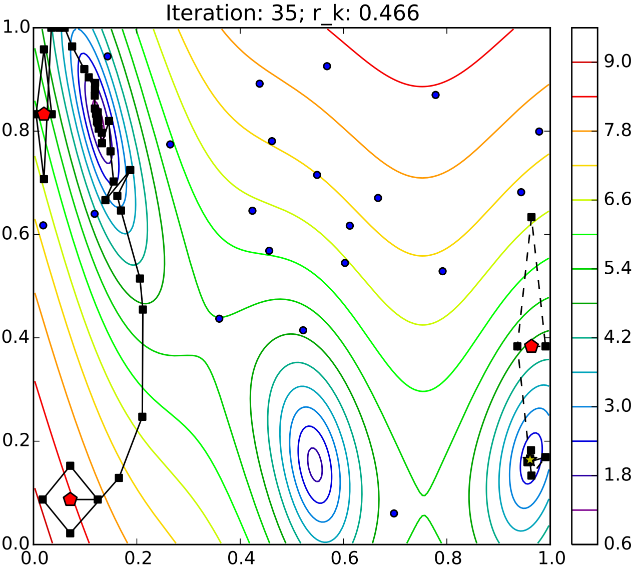
Pausing runs



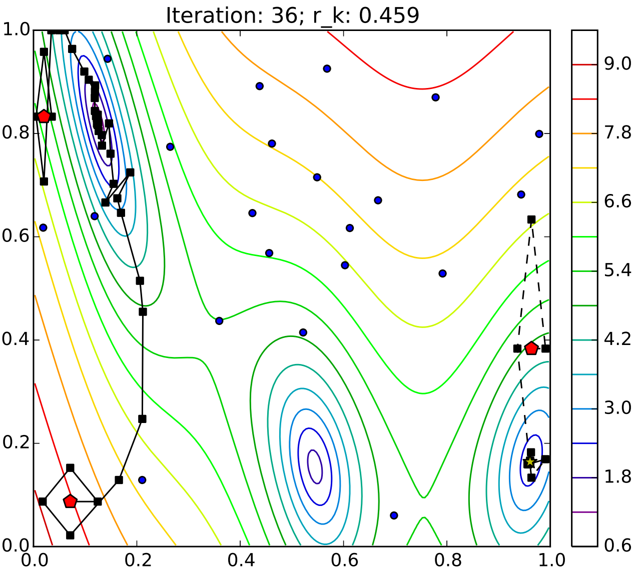
Pausing runs



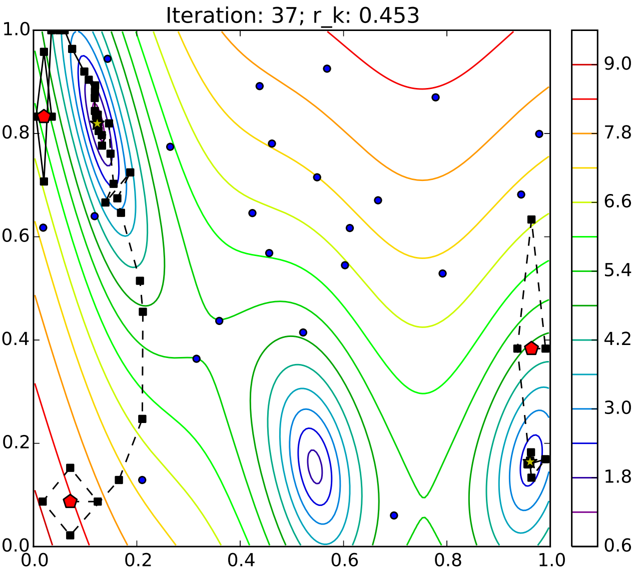
Pausing runs



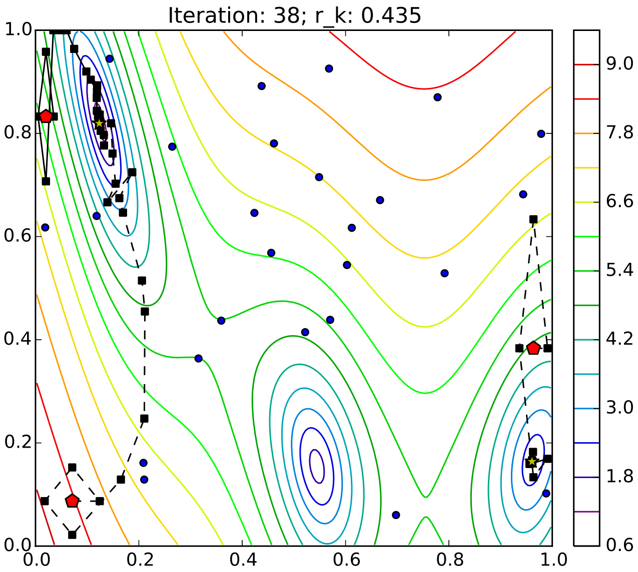
Pausing runs



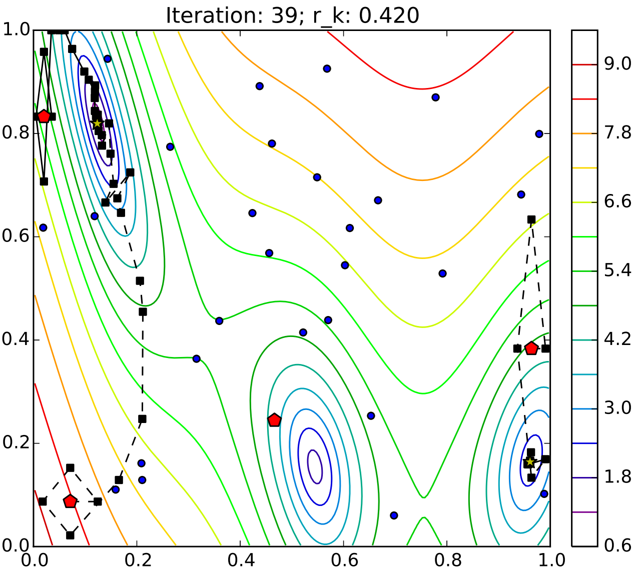
Pausing runs



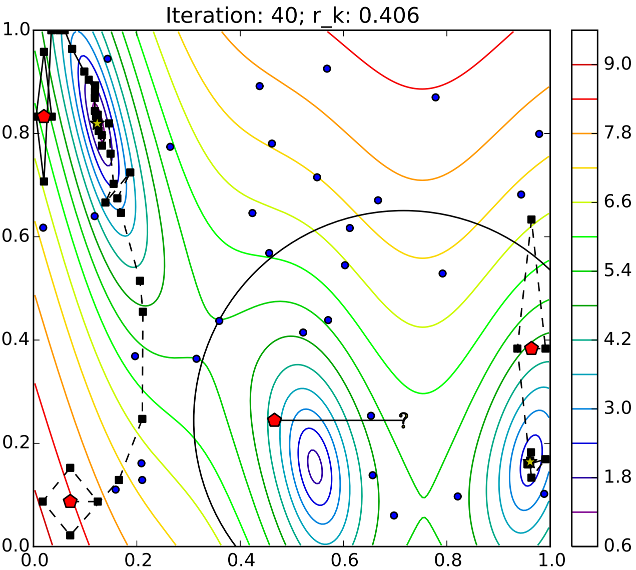
Pausing runs



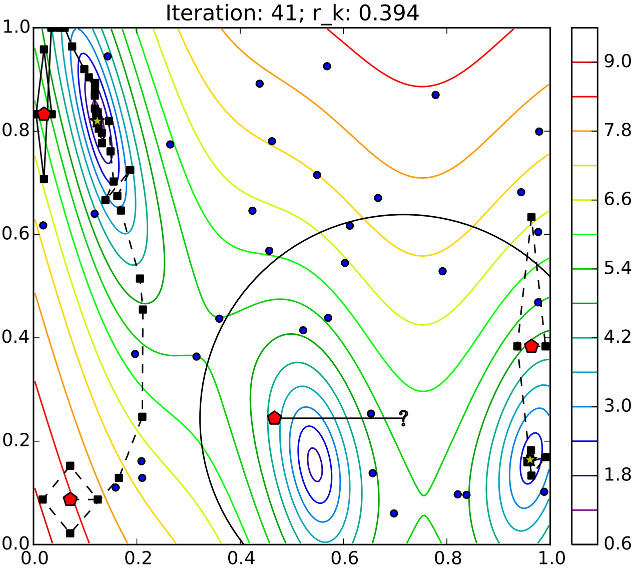
Pausing runs



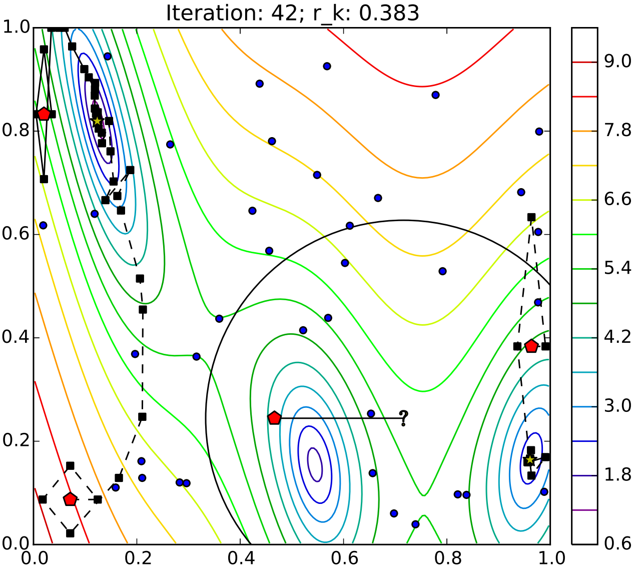
Pausing runs



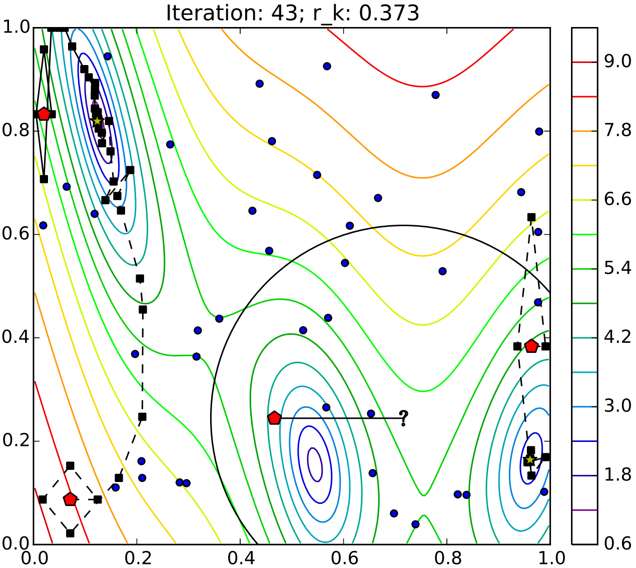
Pausing runs



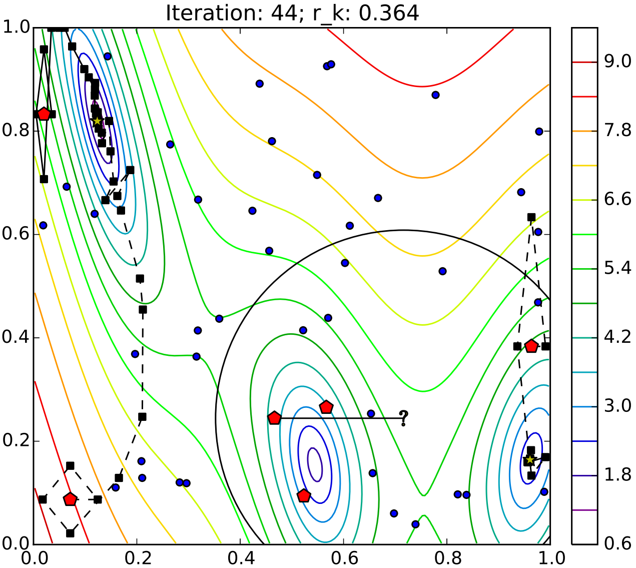
Pausing runs



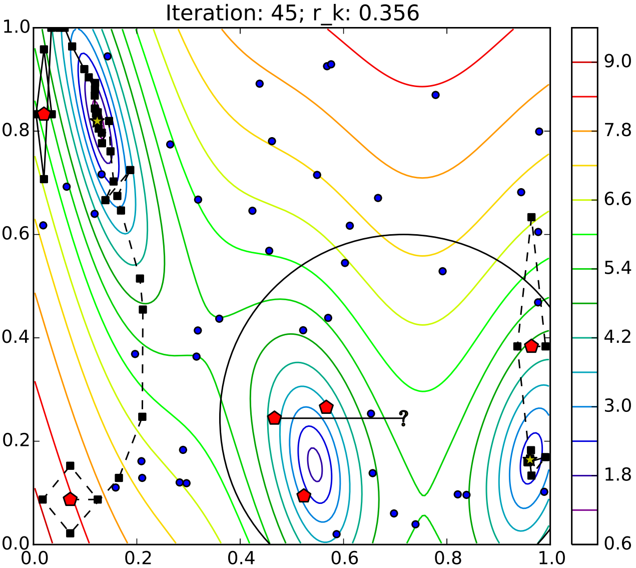
Pausing runs



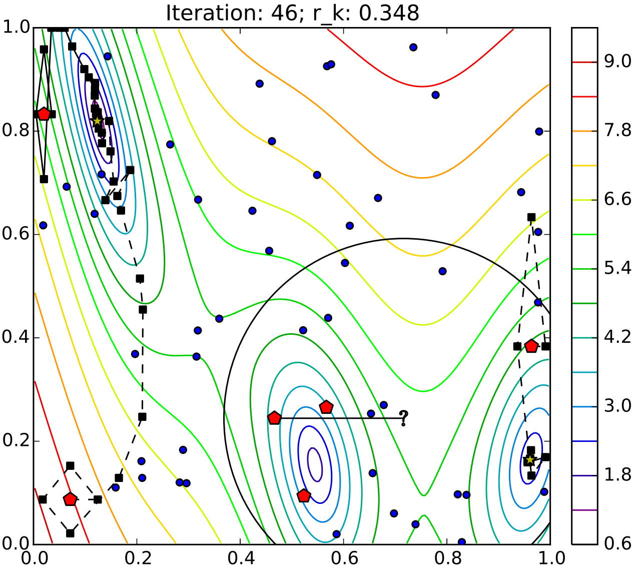
Pausing runs



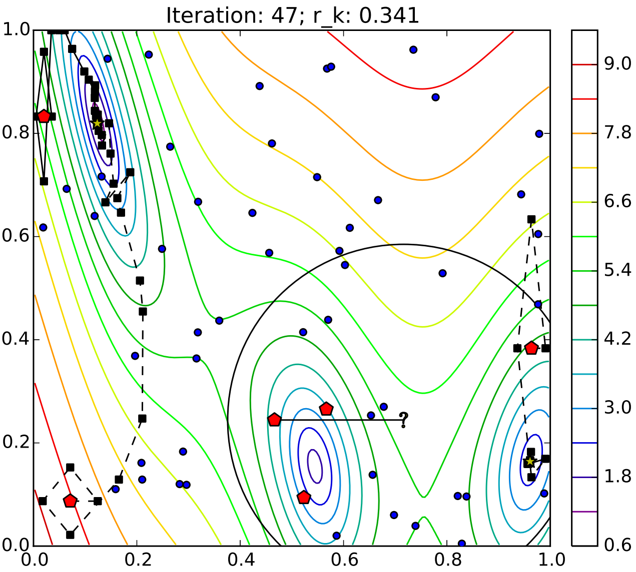
Pausing runs



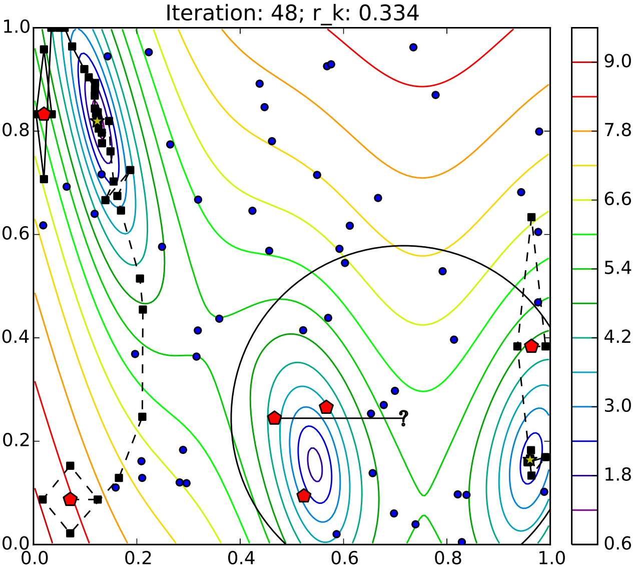
Pausing runs



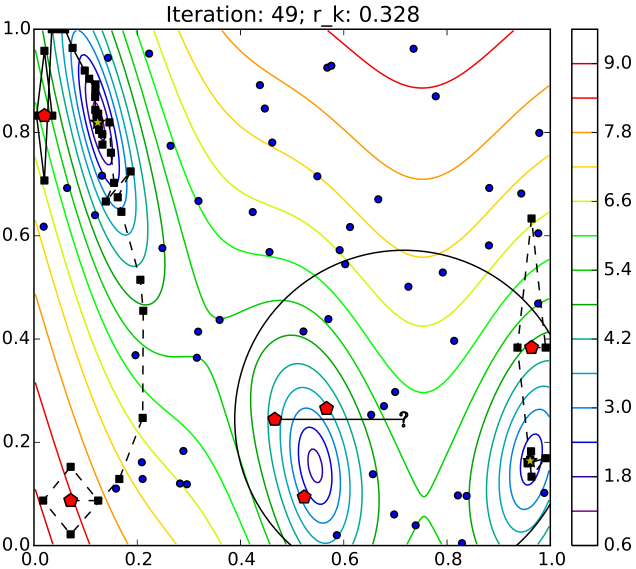
Pausing runs



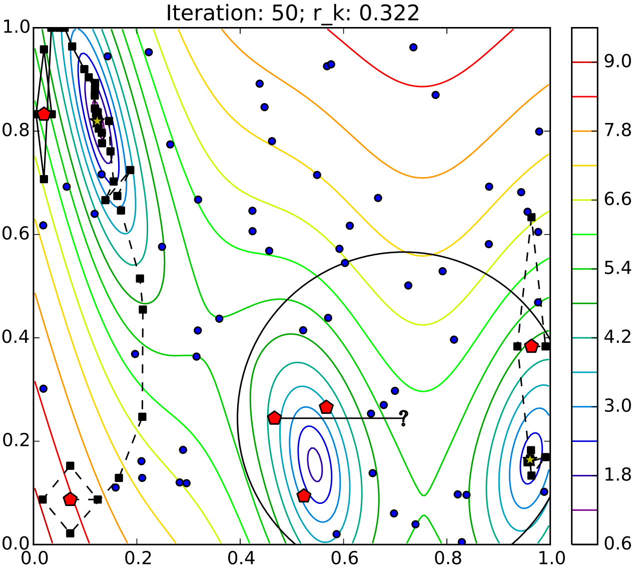
Pausing runs



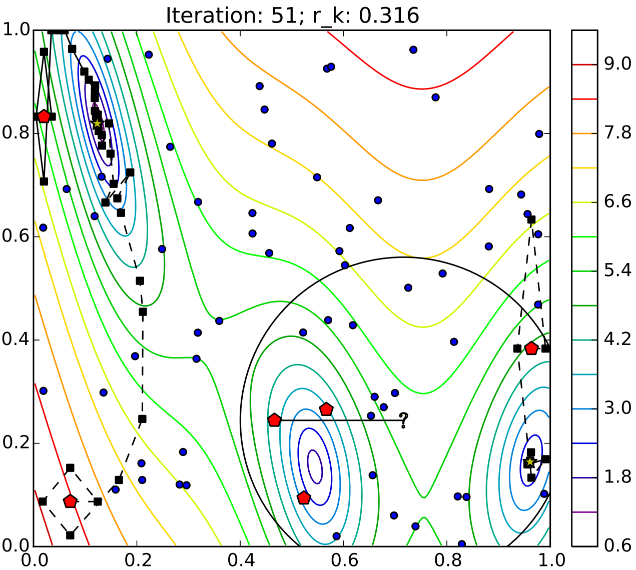
Pausing runs



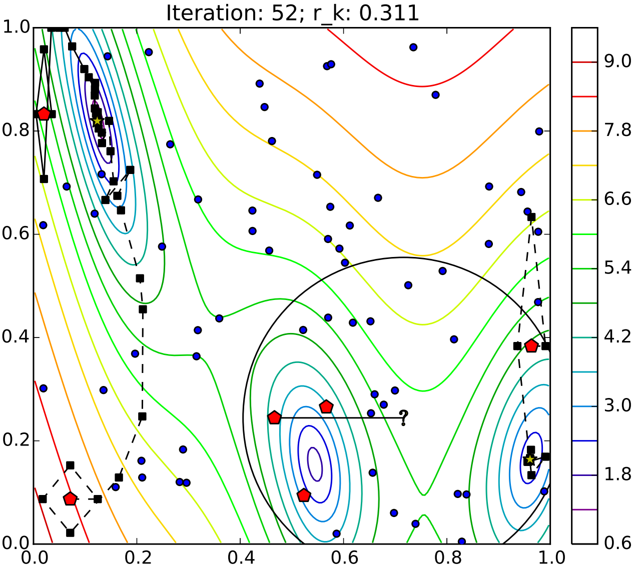
Pausing runs



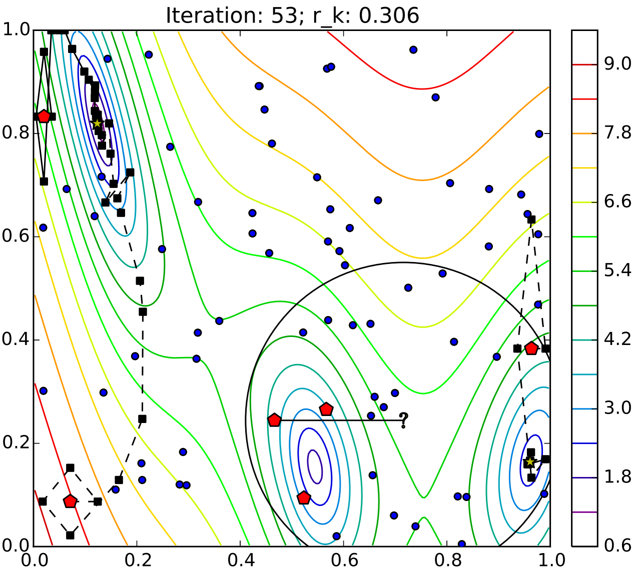
Pausing runs



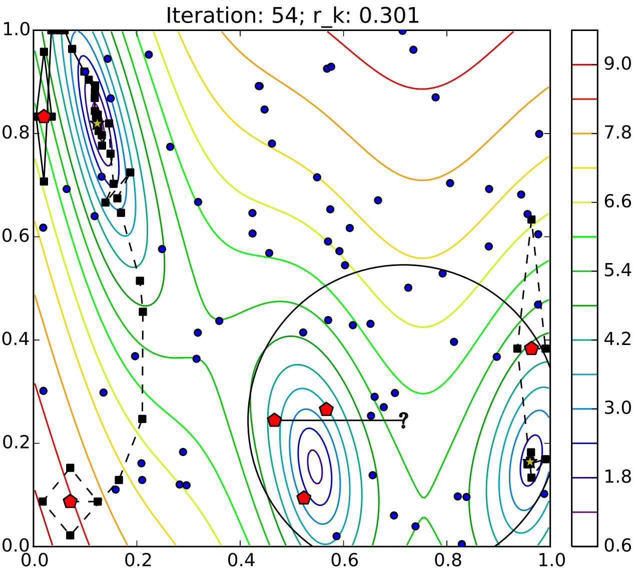
Pausing runs



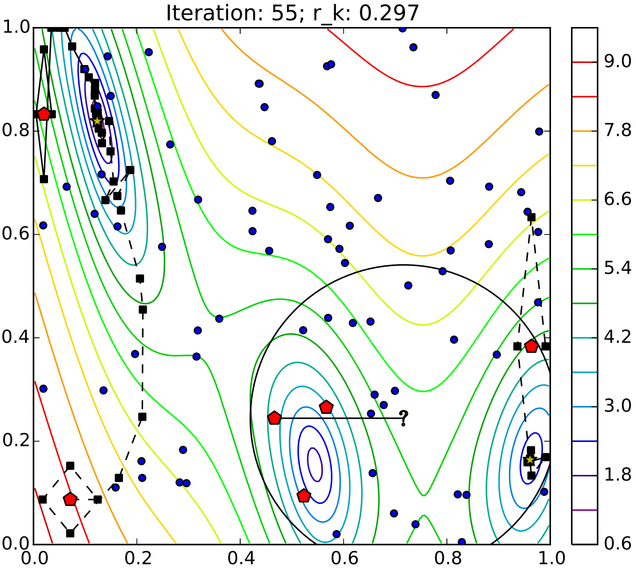
Pausing runs



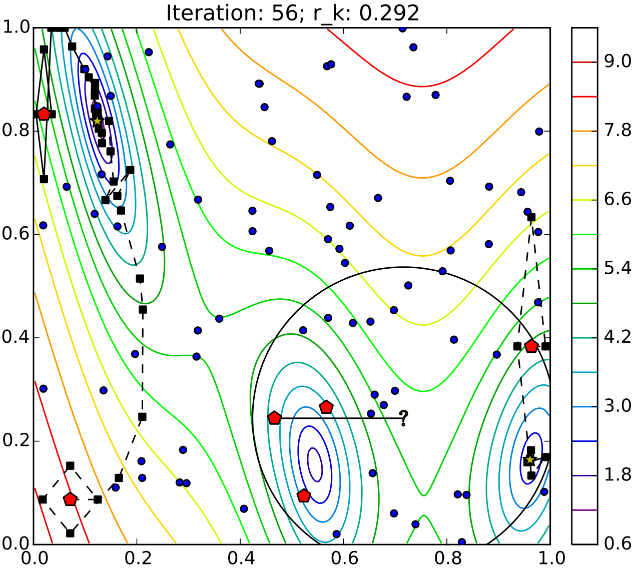
Pausing runs



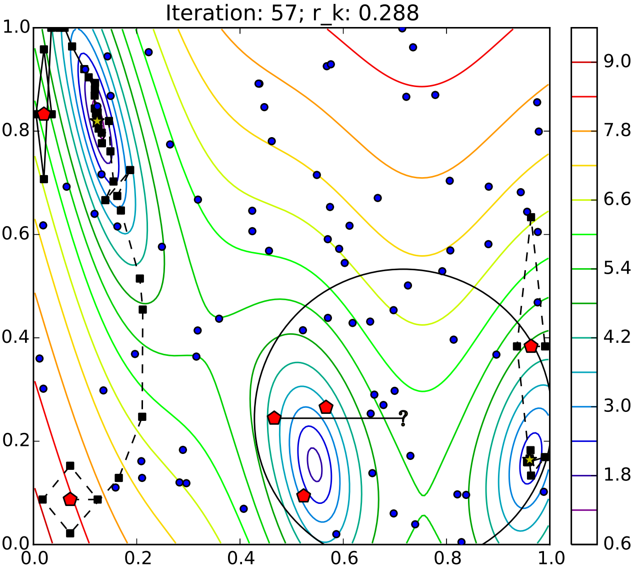
Pausing runs



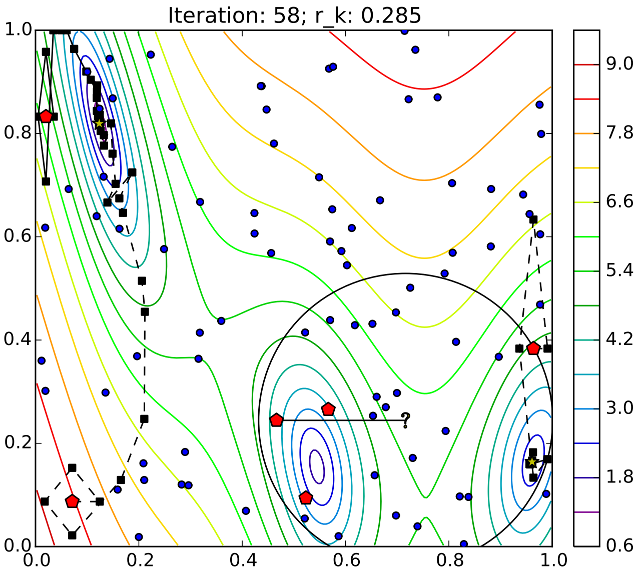
Pausing runs



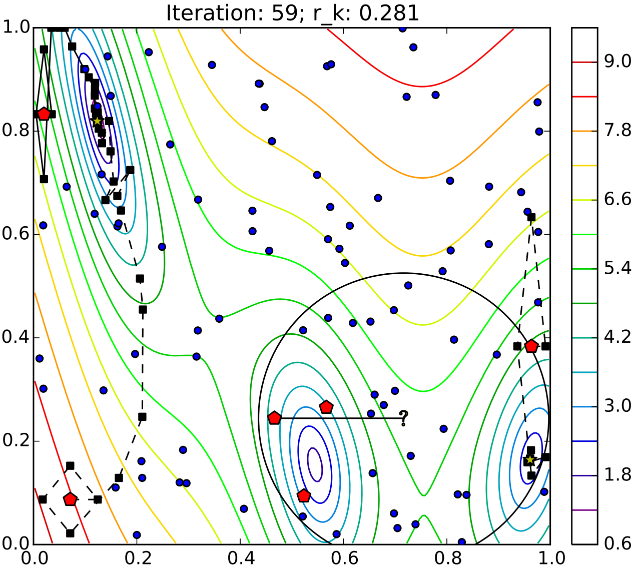
Pausing runs



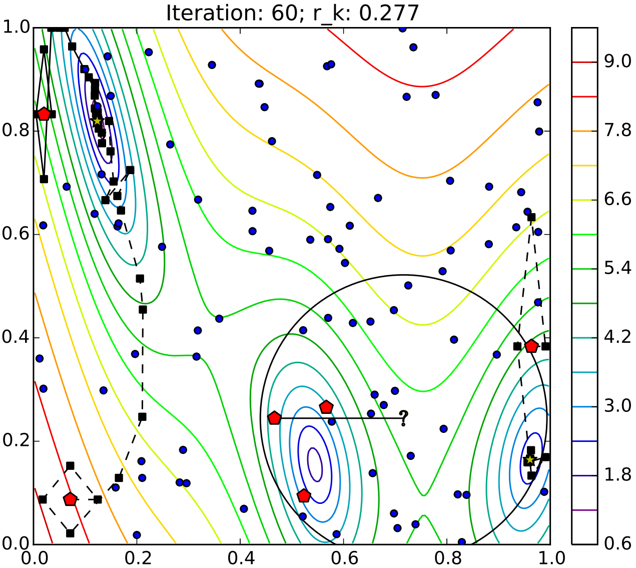
Pausing runs



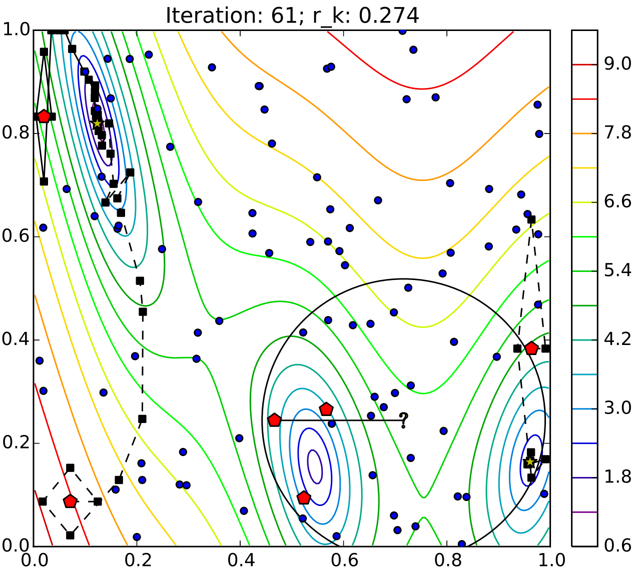
Pausing runs



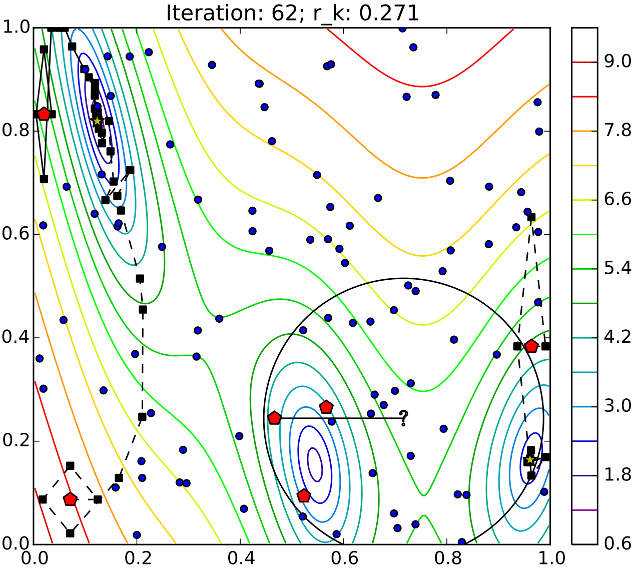
Pausing runs



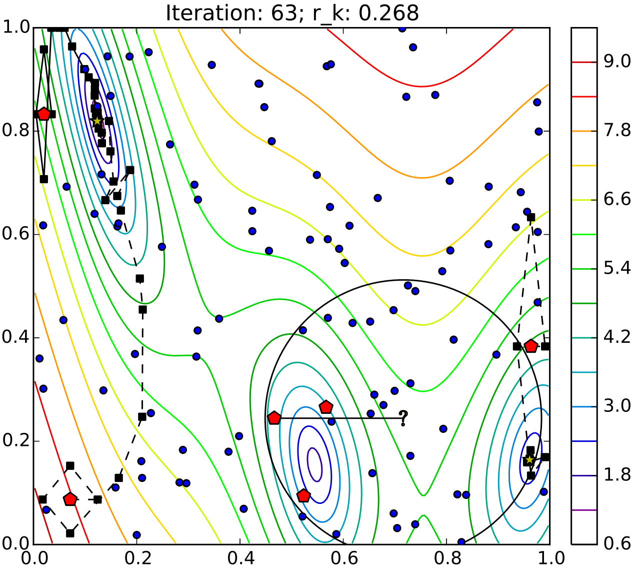
Pausing runs



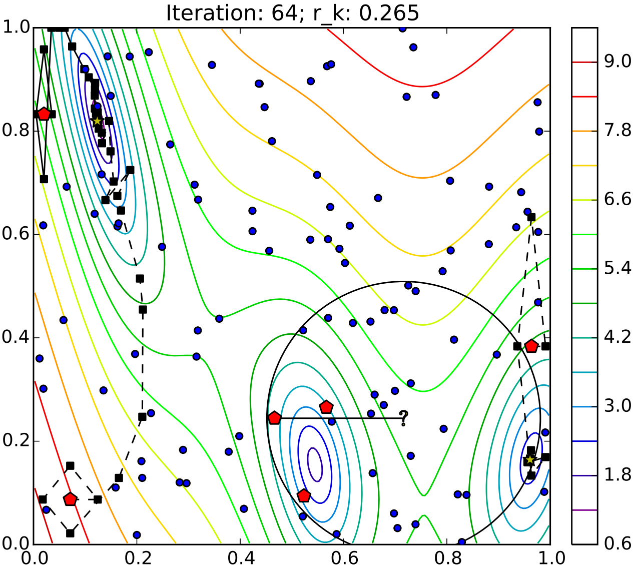
Pausing runs



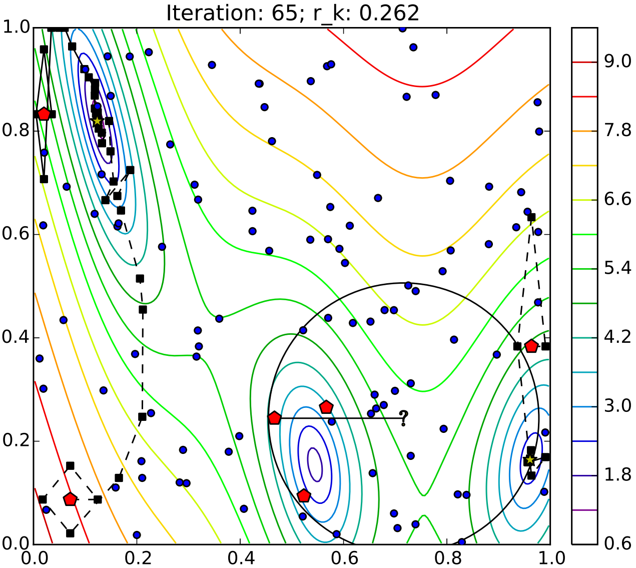
Pausing runs



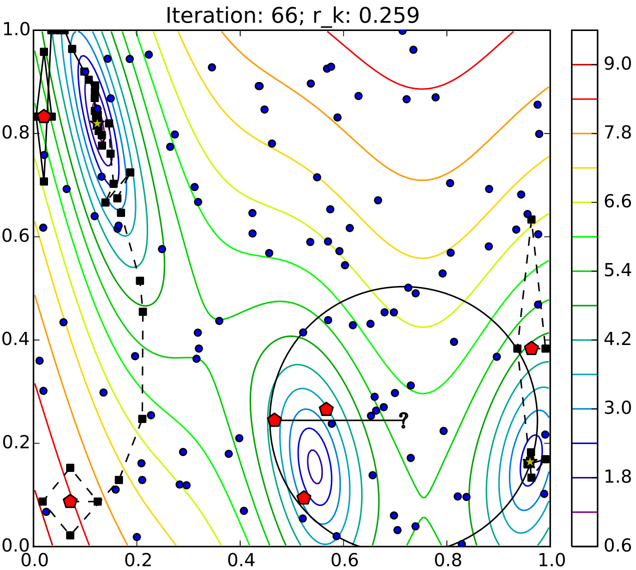
Pausing runs



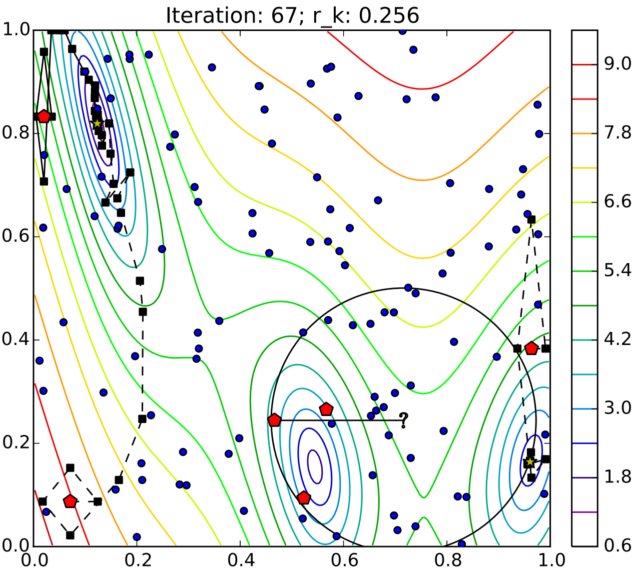
Pausing runs



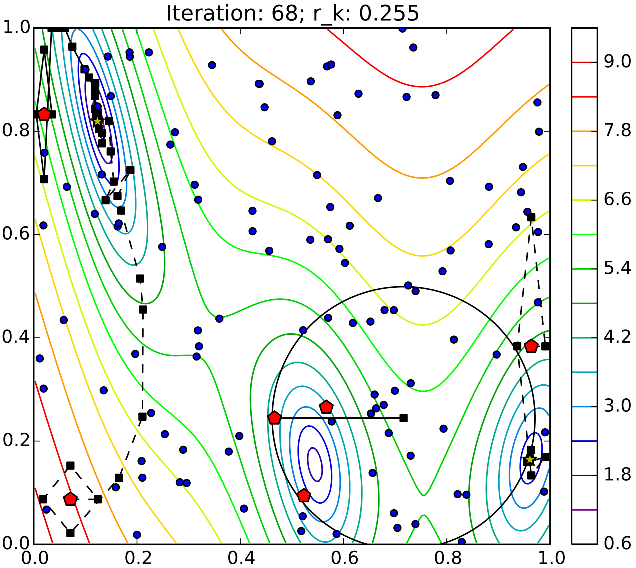
Pausing runs



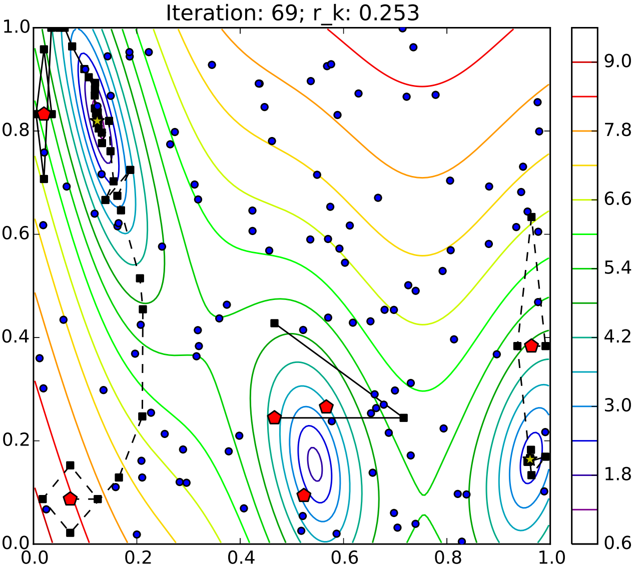
Pausing runs



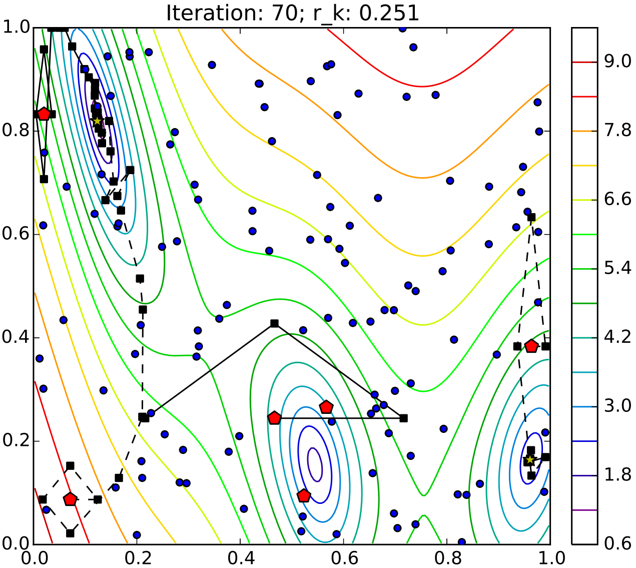
Pausing runs



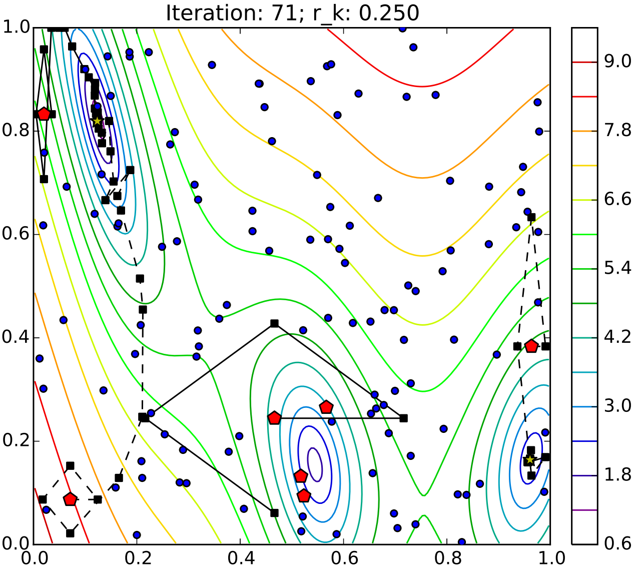
Pausing runs



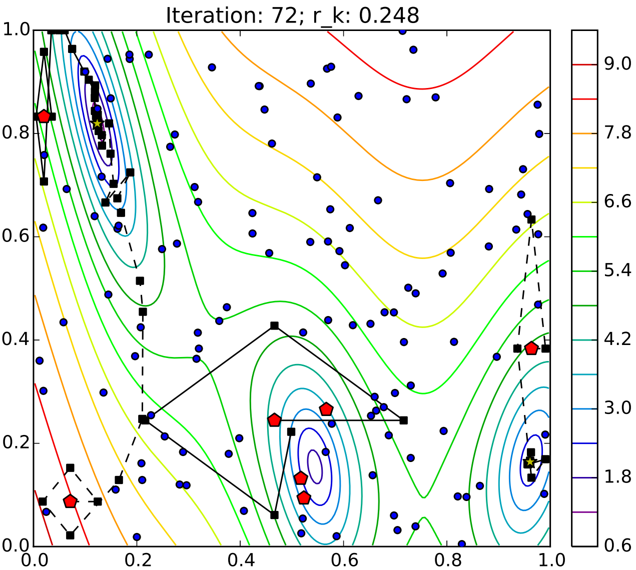
Pausing runs



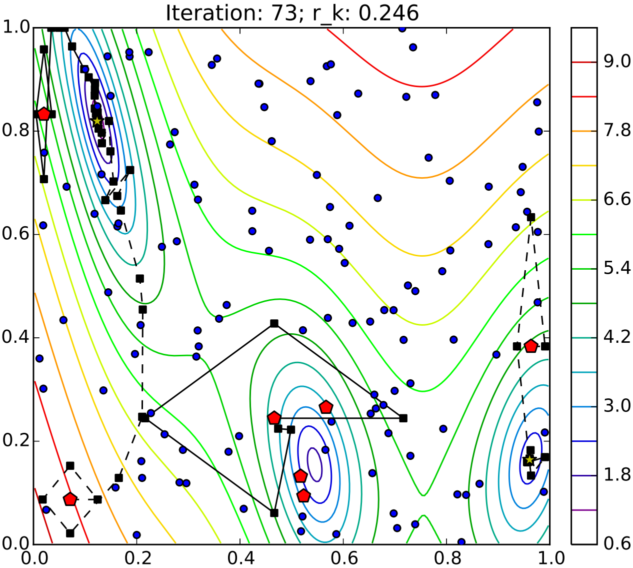
Pausing runs



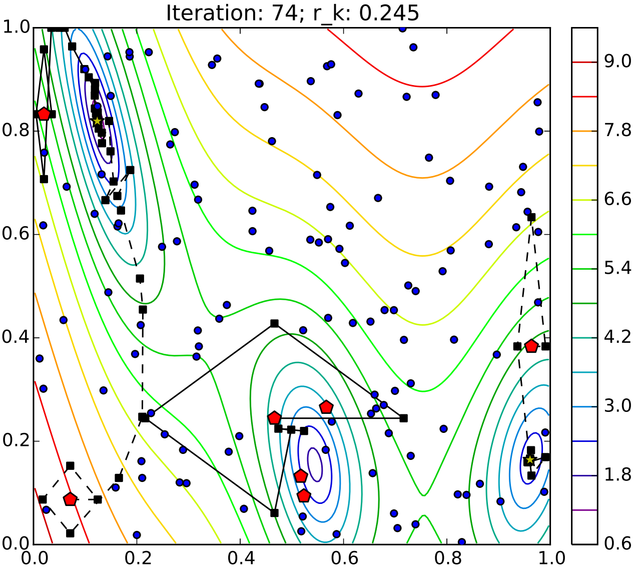
Pausing runs



Pausing runs



Pausing runs



Pausing runs